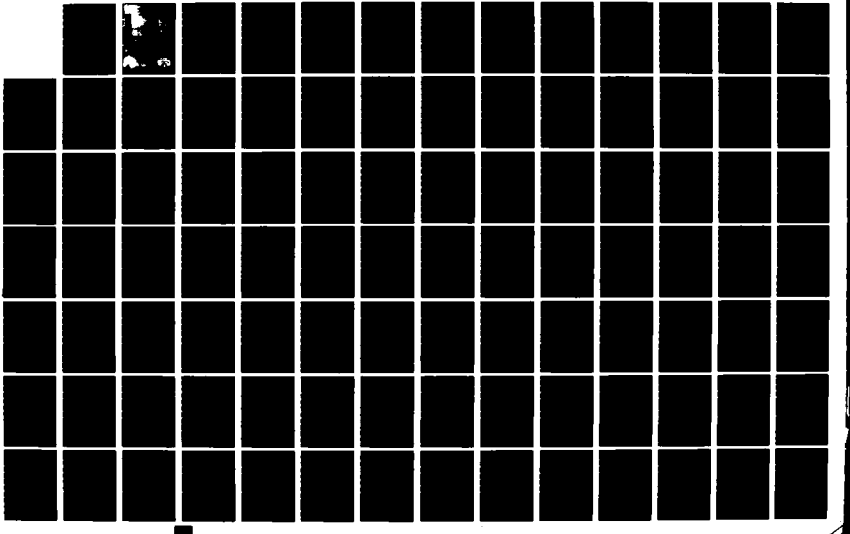
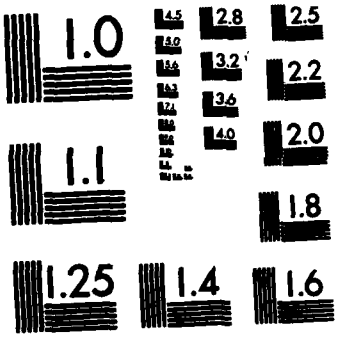
AD-A124 608 MODIFICATION OF SAVANT TO RUN ON THE PDP RSX-11M OPERATING SYSTEM(U) ROYAL AIRCRAFT ESTABLISHMENT FARNBOROUGH (ENGLAND) R J BLUFF ET AL. JAN 82 1/2

UNCLASSIFIED RAE-TM-FS(F)452 DRIC-BR-85532 F/G 9/2 NL





MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS-1963-A

AD A124608

BR85532

TECH. MEMO
PS(F) A52

[illegible] TO RUN ON THE PDP RSX-11M OPERATING SYSTEM

by

R. J. Bluff
[illegible]

January 1982

FILE COPY

83 02 01 [illegible]

UNLIMITED

ROYAL AIRCRAFT ESTABLISHMENT

Technical Memorandum FS(F) 452

Received for printing 15 January 1982

MODIFICATION OF SAVANT TO RUN ON THE PDP RSX-11M OPERATING SYSTEM

by

R. J. Bluff

P. T. May

SUMMARY

SAVANT (System Architecture Verification and Analysis Technique) is a computer program developed specifically to investigate the feasibility of an avionic system design, in terms of problems such as completeness and consistency.

The program was originally developed on a Prime 300 computer system. This Memorandum describes alterations and modifications of SAVANT, to enable it to be used on a PDP 11 RSX-11M operating system.

Copyright
©
Controller HMSO London
1982

DTIC COPY INSPECTED

Accession For	
NTIS GRA&I	☒
DTIC TAB	☐
Unannounced	☐
Justification	
By	
Distribution/	
Availability Codes	
Dist	Avail and/or Special
A	

LIST OF CONTENTS

		Page
1	INTRODUCTION	3
2	MAIN MODIFICATION TO SAVANT PROGRAM	3
3	INSTALLATION OF SAVANT PROGRAM	5
4	CONCLUSIONS	5
Acknowledgments		6
References		7
Appendices A, B, C, D, E1, E2, E3, F1, F2, F3, F4, F5, F6		
Report documentation page		inside back cover

1 INTRODUCTION

SAVANT[1] (System Architecture Verification and Analysis Technique) is a computer program which was developed specifically to investigate problems of completeness, consistency and feasibility of avionic system designs.

SAVANT was originally developed on a Prime 300 computer system and was programmed in Coral 66 using the System Designers standard SDL portable compiler. It was decided to modify the original program to run on a PDP 11/34 computer, with an RSX-11M operating system. This Memorandum describes the major program modifications to convert the original SAVANT program to Dec Coral 66 (version 3.2) and the installation of it to run under an RSX-11M operating system, see Ref 2 and 3. This Memorandum does not attempt to describe the development and application of SAVANT as Ref 1 gives an excellent detailed description.

The exercise of converting a program written in a high level language, *eg* Coral 66, from one machine to another machine has shown that it can be a tedious and time consuming process and some of the problems encountered are described below. A full listing of the program and installation files are provided to give a complete software document which will assist other users to install SAVANT, on their own machines with relative ease.

2 MAIN MODIFICATION TO SAVANT PROGRAM

The original program was developed on a Prime 300 computer using the SDL portable compiler. One of the major design aims was to make the program as transportable as possible. This aim was achieved by minimising the implementation dependent features in the main body of the code. Also, all the library calls and channel numbers for input/output were embedded in macro definitions and only the macros were called within the program. The changes to the original program fall broadly into two categories which are discussed in more detail below.

The first category includes implementation dependent differences between the two compilers. The main difference occurs in the libraries, and Appendix E.1 contains a listing of the Dec Library specification - CORIOL.CRL - required by SAVANT. The file MACDEF.COR (Appendix E.1) contains all the new macro definitions concerned with the input/output etc. All the other global macro definitions are in the file SAVGBL.COR (Appendix E.1) together with the global declarations. Wherever possible, byte arrays have replaced integer arrays in an attempt to minimise program space. The use of the character "!" as an escape character, before and after a string of layout characters, in a text string, such as:

(a) !L! - newline,

(b) !S! - space, etc

is not a feature of the Dec Coral. It was decided to edit the modification manually and this task turned out to be very tedious and with hindsight the changes should have been completed automatically with the Teco editor.

The second category of modification to the SAVANT program are as a result of the RSX-11M operating system on the PDP 11/34. The original program was too large to run as a single task, because a task is limited to a virtual address space of 32K words on the PDP 11/34. Hence, it was necessary to reduce the virtual address requirements by creating an overlaid task. Briefly, overlaying a task means it must be divided into segments; a single root segment which is always resident in memory, and a number of overlay segments which can be loaded into memory as they are required. It is not intended to discuss in detail the technique for overlaying tasks as Ref 4 provides all the relevant details. It should be stressed that the task of creating an overlay structure was considerably reduced by the fact, that the original program was designed and written in a top-down and well structured manner. In fact, the structure of the program very nearly maps directly on to the main overlay structure of the program.

The necessity to produce an overlaid task also provided an opportunity to incorporate all program modules into a library called SAVLIB. The library feature results in a considerable saving of time when program modification and maintenance are necessary. It allows single modules to be compiled and inserted into the library without the need to recompile the complete program. This is useful because, if, for example a global parameter is to be changed then it will usually be safer to recompile the whole program, which could take several hours. To assist library building and overlay construction every SAVANT procedure was copied into an individual file. These files were given an identical name to the corresponding procedure name. The procedure names were also truncated into a six letter combination, to ensure uniqueness of module and module entry names at task build time. Appendix A contains cross-reference tables between the old SAVANT procedure and the new file/procedure names. The first table lists the new six letter file/procedure name in column 1 ; column 3 is the corresponding SAVANT command name where it is appropriate; column 4 is the old SAVANT procedure name; and column 5 is the program level, where level 0 is the top level, *eg* SAV2MN. The other tables are similar; the first column contains new file/procedure names and old procedure names, in alphabetical order, respectively.

All the procedures are in separate files which have the same names as the procedure and these files are then placed in either an OVL or LIB file together with the four global files (*viz* CORIOL.CRL, PROCS,COR, SAVGBL.COR, and MACDEF.COR) using the Coral 'INCLUDE' facility. These files are the basic program modules, which are used to construct SAVLIB and they are compiled as individual Coral segments.

Appendix C contains examples of LIB and OVL files respectively. The first file is LIBOO.COR and includes the procedure file ACTPRC.COR and is headed 'Coral' MODOO. Appendix D contains a cross reference of LIB/MOD and OVL/NUM with a segment name which is the *same as* the procedure filename. The OVL files contain the procedure which correspond to the SAVANT commands and the LIB files contain the remainder of the procedures.

Appendix E.1 contains the listings of the four global files. Appendix B illustrates the excellent structure of the original SAVANT program, and is a cross reference

table of procedure calls versus program levels. The level 0 and 1 are excluded for the sake of clarity and only consist of SAV2MN at level 0 and SETCMD and OBEYCM at level 1. Consider the command CDR which is called from the main program SAV2MN at level 0 via OBEYCM at level 1. This in turn calls the procedure CHDART at level 2. CHDART calls CMDWRN and CHDAV at level 3. CHDAV calls GETNV at level 4 and LOCNAM at level 5, while LOCNAM calls FINNAM at level 6.

The file handling differences between the two operating systems necessitated extensive modification to the old 'FIND FILE' procedure and the substitution of two new procedures to replace the old 'READ VALID FILENAME'. Also two other procedures CMPSTR and GETNUM were required and the listing of all these procedures is given in Appendix E.2 together with the remainder of the procedures in Appendix E.3.

The original procedure 'OBEY COMMAND' caused the operating system to abort the Dec Coral compiler task and was modified. The Dec compiler is not capable of handling the long " 'IF' condition 'THEN' consequence 'ELSE' .... " conditional statement.

This long statement, was reconstructed into three shorter conditional statements by the introduction of compound statements, see listing of procedure OBEYCM.COR in Appendix E.2.

The use of byte arrays introduced several minor errors at run time and these were overcome by ensuring that byte arrays started on word boundaries.

3 INSTALLATION OF SAVANT PROGRAM

The installation of SAVANT is straight forward and is accomplished by running the indirect command file SAVMN.CMD listed in Appendix F.6. This command file expects to find all files in SY: and in the current default UIC. The main command file calls compilation indirect command files OVLCMP.CMD (Appendix F.3) and LIBCMP.CMD (Appendix F.2); it then builds the SAVANT library file SAVLIB.OLB with LIBGEN.CMD (Appendix F.4) and finally runs the task builder using indirect command file SAVBLD.CMD (Appendix F.5) which requires the overlay descriptor file SAVANT.ODL (Appendix F.1).

4 CONCLUSIONS

It was assumed that it would be a fairly straight forward exercise to transfer a well structured program written in Coral from the Prime 300 to the PDP 11/34 as both machines had Coral compilers. In practice it turned out to be a time consuming exercise. This Memorandum describes the major modification necessary as a result of the differences between the compilers and operating systems, on the two machines. The technique of partitioning a large program into several individual segments which are then compiled separately and installed into a library has proved a well worthwhile feature. It has saved a considerable amount of time during program development and has introduced flexibility into the software which assists task building of overlaid tasks. The program has now been running for several weeks on the PDP, but has only been subjected to fairly limited use. It is impossible to guarantee that the program is free from error and it is hoped that any errors found by other users will be reported to the author. It is also intended to run this program on a Dec VAX/11-750.

Acknowledgments

The author wishes to thank Dr N. Tatham and Dr J.M. Barrett of Flight Systems Department, for the initial part of the conversion exercise.

REFERENCES

No.	Author	Title, etc
1	A.A. Callaway	SAVANT - a database manipulation program for system architecture verification and design analysis. RAE Technical Report 80101 (1981)
2	Dec	IAS/RSX/UMS Coral 66 users guide; AA-D112A-TK.
3	Dec	Coral 66 language reference manual; AD-A11A-TC.
4	Dec	RSX-11M-plus task builder manual; AA-H266A-TC.

APPENDIX A

CROSS REFERENCE LISTS

LIST OF FILENAMES AND PROCEDURE NAMES BY LEVEL NUMBER

File	No	CMD	Procedure name	Level
SAV2MN	1		MAIN PROGRAM	0
SETCMD	2		SET COMMAND STRINGS	1
OBEYCM	3		OBEY COMMAND	1
RESET	4	RSP	RESET PARAMETERS	2
INTTYE	5	ITE	INPUT TERMINAL ENTRIES	2
INFILE	6	IFE	INPUT FILED ENTRIES	2
INFILM	7	IFM	INPUT FILED MESSAGES	2
LISENT	8	LEN	LIST ENTRIES	2
LISSSD	9	LSD	LIST SS DATA	2
LISSSN	10	LSN	LIST SS NAMES	2
LISDNM	11	LDN	LIST DATA NAMES	2
TRDATA	12	TDP	TRACE DATA PATH	2
CHSSN	13	CSN	CHANGE SS NAME	2
CHDNM	14	CDN	CHANGE DATA NAME	2
CHRATE	15	CRE	CHANGE RATE ENTRY	2
CHRATG	16	CRG	CHANGE RATE GENERAL	2
CHDART	17	CDR	CHANGE DATA RATE	2
CHSSRT	18	CSR	CHANGE SS RATE	2
CHPRCE	19	CPE	CHANGE PREC ENTRY	2
CHPRCG	20	CPG	CHANGE PREC GENERAL	2
CHDAPR	22	CDP	CHANGE DATA PREC	2
CHSSPR	23	CSP	CHANGE SS PREC	2
CHUNIE	24	CUE	CHANGE UNITS ENTRY	2
CHUNIG	25	CUG	CHANGE UNITS GENERAL	2
CHDAUN	26	CDU	CHANGE DATA UNITS	2
DELDAR	27	DDR	DELETE DATA REFERENCE	2
DEL1EN	28	DOE	DELETE ONE ENTRY	2
FILENT	29	FEN	FILE ENTRIES	2
FILESD	30	FSD	FILE SUBSYSTEM DATA	2
CLRDB	31	CLR	CLEAR DATABASE	2
CONFIG	32	CFS	CONFIGURE SYSTEM	2
LISDAP	33	LDP	LIST DATA PATHS	2
LISCOS	34	LCS	LIST CONFIGURED SUBSYSTEMS	2
LISDAT	35	LDT	LIST DATA TRAFFIC	2
MAPDAT	36	MDT	MAP DATA TRAFFIC	2
SSPRDA	37	SSP	SOURCE SINK PAIR DATA	2
CKCONS	38	CCS	CHECK CONSISTENCY	2
CKCORR	39	CCR	CHECK CORRELATION	2
LISMES	40	LBM	LIST BUS MESSAGES	2
LISDDA	41	LDD	LIST DIRECT DATA	2
CALLOA	42	CLD	CALCULATE LOADINGS	2
SUMMES	43	SUM	SUMMARISE MESSAGES	2
LIS1ME	44	LOM	LIST ONE MESSAGE	2
SETRET	45	SRC	SET RETRY CODE	2
SETTAD	46	STA	SET TERMINAL ADDRESS	2
LISSUB	47	LSS	LIST SUBSETS	2
REORDM	48	ROM	REORDER MESSAGE	2
REFMTM	49	RFM	REFORMAT MESSAGES	2

FILMES	50	FLM	FILE MESSAGES	2
DISMAN	51	DMS	DISMANTLE SYSTEM	2
FRMSYS	52	FMS	FORM SYSTEM	2
LISSAD	53	LSA	LIST SUBADDRESSES	2
FILSCH	54	FLS	FILE SCHEDULES	2
UNFSYS	55	UFS	UNFORM SYSTEM	2
INIT	56		INITIALISE	3
CMDWRN	57		COMMAND WARN	3
INTPCH	58		INT PARAM CHANGE	3
FLOPCH	59		FLO PARAM CHANGE	3
INDAT	60		INPUT DATA	3
INNAM	61		INPUT NAMELIST	3
LISREF	62		LIST REFLIST	3
LISNAM	63		LIST NAMES	3
CHNAM	64		CHANGE NAME	3
CHNUME	65		CHANGE NUMERIC ENTRY	3
GENNCH	66		GENERAL NUMBER CHANGE	3
CHDAV	67		CHANGE DATA VALUE	3
CHSSV	68		CHANGE SS VALUE	3
DELENT	69		DELETE ENTRY	3
FILREF	70		FILE REFLIST	3
FRMTXS	71		FORM TX SS	3
FRMTXD	72		FORM TX DATA	3
FRMRXS	73		FORM RX SS	3
FATLCK	74		FATAL CHECK	3
FRMMES	75		FORM MESSAGES	3
FINBSS	76		FIND BUS SS	3
PARNUM	77		PARTITION NUMBER	3
SETMAP	78		SET MAP	3
LISSSP	79		LIST SS PAIR	3
FINMES	80		FIND MESSAGE	3
CKTA	81		CHECK TA	3
LISMDA	82		LIST MESSAGE DATA	3
FILNAM	83		FILE NAMELIST	3
FINCOD	84		FIND CODE	3
FRMSUB	85		FORM SUBSETS	3
OPELIS	86		OPEN LIST	3
CLOLIS	87		CLOSE LIST	3
CKTAST	88		CHECK TA SET	3
FRMSAD	89		FORM SUBADDRESSES	3
FILTAB	90		FILE TABLES	3
FILSAD	91		FILE SUBADDRESSES	3
LISCMD	92		LIST COMMANDS	4
FINFIL	93		FIND FILE	4
CLRCFL	94		CLEAR CF LISTS	4
CLRMEL	95		CLEAR MESSAGE LISTS	4
OPTION	96		OPTION	4
READEN	97		READ ENTRY	4
CONVNM	98		CONVERT NAME	4
GETNV	99		GET NEW VALUE	4
FINENT	100		FIND ENTRY	4
CKREF	101		CHECK REFERENCE	4
GET OLV	102		GET OLD VALUE	4
CHK #	10[illegible]		CHANGE NUMBER	4
WORDS	10[illegible]		WORDS	4
FATLPR	105		FATAL PRINT	4
SENMES	106		SEND MESSAGE	4
CKSUB	107		CHECK SUBSET	4
FINBC	108		FIND BUSCON	4
FILCW	109		FILE CW	4

FIL1SA	110	FILE ONE SA	4
FILDIR	111	FILE DIRECTS	4
RECNAM	112	RECORD NAME	5
GETTR	113	GET TR	5
LOCNAM	114	LOCATE NAME	5
DELNAM	115	DELETE NAME	5
REPOIN	116	REPOINT	5
SENCON	117	SEND CONTENTS	5
FILHDR	118	FILE HEADER	5
FINNAM	119	FIND NAME	6
MOVSTR	120	MOVE STRING	6
ACTPRC	121	ACTUAL PREC	6
CLISTR	122	CLISTREF	6
FILNUM	123	FILE NUMBER	6
OPFIRE	124		6
OPFIWR	125		6

LIST OF FILENAMES AND PROCEDURE NAMES BY FILENAME

File	No	CMD	Procedure name	Level
ACTPRC	121		ACTUAL PREC	6
CALLOA	42	CLD	CALCULATE LOADINGS	2
CHDAPR	22	CDP	CHANGE DATA PREC	2
CHDART	17	CDR	CHANGE DATA RATE	2
CHDAUN	26	CDU	CHANGE DATA UNITS	2
CHDAV	67		CHANGE DATA VALUE	3
CHDNM	14	CDN	CHANGE DATA NAME	2
CHNAM	64		CHANGE NAME	3
CHNUM	103		CHANGE NUMBER	4
CHNUME	65		CHANGE NUMERIC ENTRY	3
CHPRCE	19	CPE	CHANGE PREC ENTRY	2
CHPRCG	20	CPG	CHANGE PREC GENERAL	2
CHRATE	15	CRE	CHANGE RATE ENTRY	2
CHRATG	16	CRG	CHANGE RATE GENERAL	2
CHSSN	13	CSN	CHANGE SS NAME	2
CHSSPR	23	CSP	CHANGE SS PREC	2
CHSSRT	18	CSR	CHANGE SS RATE	2
CHSSV	68		CHANGE SS VALUE	3
CHUNIE	24	CUE	CHANGE UNITS ENTRY	2
CHUNIG	25	CUG	CHANGE UNITS GENERAL	2
CKCONS	38	CCS	CHECK CONSISTENCY	2
CKCORR	39	CCR	CHECK CORRELATION	2
CKREF	101		CHECK REFERENCE	4
CKSUB	107		CHECK SUBSET	4
CKTA	81		CHECK TA	3
CKTAST	88		CHECK TA SET	3
CLISTR	122		CLISTREF	6
CLOLIS	87		CLOSE LIST	3
CLRCFL	94		CLEAR CF LISTS	4
CLRDB	31	CLR	CLEAR DATABASE	2
CLRMEL	95		CLEAR MESSAGE LISTS	4
CMDWRN	57		COMMAND WARN	3
CONFIG	32	CFS	CONFIGURE SYSTEM	2
CONVNM	98		CONVERT NAME	4
DEL1EN	28	DOE	DELETE ONE ENTRY	2
DELDAR	27	DDR	DELETE DATA REFERENCE	2
DELENT	69		DELETE ENTRY	3
DELNAM	115		DELETE NAME	5
DISMAN	51	DMS	DISMANTLE SYSTEM	2
FATLCK	74		FATAL CHECK	3
FATLPR	105		FATAL PRINT	4
FIL1SA	110		FILE ONE SA	4
FILCW	109		FILE CW	4
FILDIR	111		FILE DIRECTS	4
FILENT	29	FEN	FILE ENTRIES	2
FILESD	30	FSD	FILE SUBSYSTEM DATA	2
FILHDR	118		FILE HEADER	5
FILMES	50	FLM	FILE MESSAGES	2
FILNAM	83		FILE NAMELIST	3
FILNUM	124		FILE NUMBER	6
FILREF	70		FILE REFLIST	3
FILSAD	91		FILE SUBADDRESSES	3
FILSCH	54	FLS	FILE SCHEDULES	2
FILTAB	90		FILE TABLES	3
FINBC	108		FIND BUSCON	4
FINBSS	76		FIND BUS SS	3
FINCOD	84		FIND CODE	3

FINENT	100		FIND ENTRY	4
FINFIL	93		FIND FILE	4
FINMES	80		FIND MESSAGE	3
FINNAM	119		FIND NAME	6
FLOPCH	59		FLO PARAM CHANGE	3
FRMMES	75		FORM MESSAGES	3
FRMRXS	73		FORM RX SS	3
FRMSAD	89		FORM SUBADDRESSES	3
FRMSUB	85		FORM SUBSETS	3
FRMSYS	52	FMS	FORM SYSTEM	2
FRMTXD	72		FORM TX DATA	3
FRMTXS	71		FORM TX SS	3
GENNCH	66		GENERAL NUMBER CHANGE	3
GETNV	99		GET NEW VALUE	4
GETOLV	102		GET OLD VALUE	4
GETTR	113		GET TR	5
INDAT	60		INPUT DATA	3
INFILE	6	IFE	INPUT FILED ENTRIES	2
INFILM	7	IFM	INPUT FILED MESSAGES	2
INIT	56		INITIALISE	3
INNAM	61		INPUT NAMELIST	3
INTPCH	58		INT PARAM CHANGE	3
INTTYE	5	ITE	INPUT TERMINAL ENTRIES	2
LIS1ME	44	LOM	LIST ONE MESSAGE	2
LISCMD	92		LIST COMMANDS	4
LISCOS	34	LCS	LIST CONFIGURED SUBSYSTEMS	2
LISDAP	33	LDP	LIST DATA PATHS	2
LISDAT	35	LDT	LIST DATA TRAFFIC	2
LISDDA	41	LDD	LIST DIRECT DATA	2
LISDNM	11	LDN	LIST DATA NAMES	2
LISENT	8	LEN	LIST ENTRIES	2
LISMDA	82		LIST MESSAGE DATA	3
LISMES	40	LBM	LIST BUS MESSAGES	2
LISNAM	63		LIST NAMES	3
LISREF	62		LIST REFLIST	3
LISSAD	53	LSA	LIST SUBADDRESSES	2
LISSSD	9	LSD	LIST SS DATA	2
LISSSN	10	LSN	LIST SS NAMES	2
LISSSP	79		LIST SS PAIR	3
LISSUB	47	LSS	LIST SUBSETS	2
LOCNAM	114		LOCATE NAME	5
MAPDAT	36	MDT	MAP DATA TRAFFIC	2
MOVSTR	120		MOVE STRING	6
OBEYCM	3		OBEY COMMAND	1
OPELIS	86		OPEN LIST	3
OPTION	96		OPTION	4
READEN	97		READ ENTRY	4
RECNAM	112		RECORD NAME	5
REFMTM	49	RFM	REFORMAT MESSAGES	2
REORDM	48	ROM	REORDER MESSAGE	2
REPOIN	116		REPOINT	5
RESET	4	RSP	RESET PARAMETERS	2
SAV2MN	1		MAIN PROGRAM	0
SENCON	117		SEND CONTENTS	5
SENMES	106		SEND MESSAGE	4
SETCMD	2		SET COMMAND STRINGS	1
SETMAP	78		SET MAP	3
SETRET	45	SRC	SET RETRY CODE	2
SETTAD	46	STA	SET TERMINAL ADDRESS	2
PARNUM	77		PARTITION NUMBER	3
SSPRDA	37	SSP	SOURCE SINK PAIR DATA	2
SUMMES	43	SUM	SUMMARISE MESSAGES	2

TRDATA	12	TDP	TRACE DATA PATH	2
UNFSYS	55	UFS	UNFORM SYSTEM	2
WORDS	104		WORDS	4

LIST OF FILENAMES AND PROCEDURE NAMES BY PROCEDURE NAME

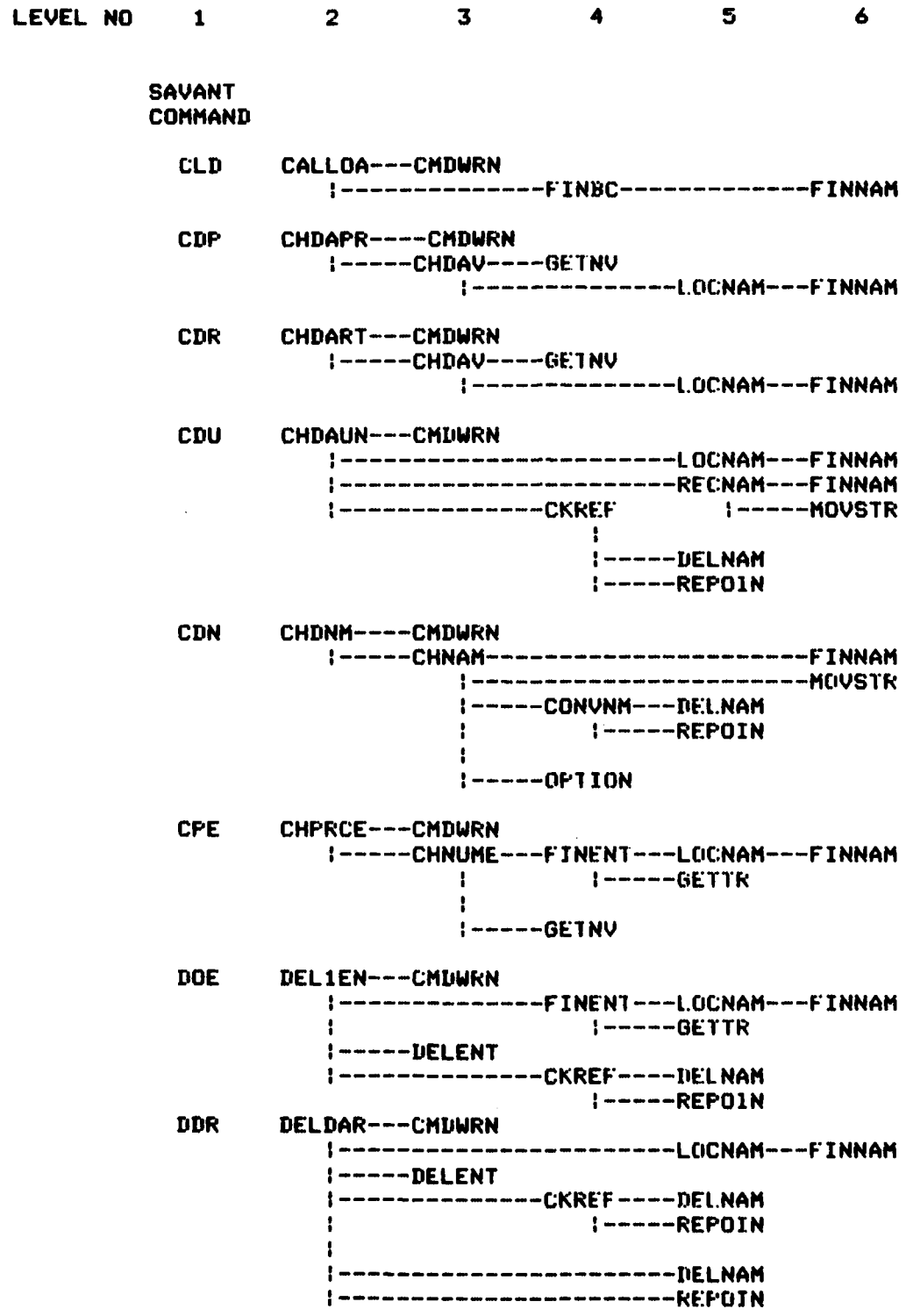
Procedure name	Level	File	No	CMD
ACTUAL PREC	6	ACTPRC	121	
CALCULATE LOADINGS	2	CALLOA	42	CLD
CHANGE DATA NAME	2	CHDNM	14	CDN
CHANGE DATA PREC	2	CHDAPR	22	CDP
CHANGE DATA RATE	2	CHDART	17	CDR
CHANGE DATA UNITS	2	CHDAUN	26	CDU
CHANGE DATA VALUE	3	CHDAV	67	
CHANGE NAME	3	CHNAM	64	
CHANGE NUMBER	4	CHNUM	103	
CHANGE NUMERIC ENTRY	3	CHNUME	65	
CHANGE PREC ENTRY	2	CHPRCE	19	CPE
CHANGE PREC GENERAL	2	CHPRCG	20	CPG
CHANGE RATE ENTRY	2	CHRATE	15	CRE
CHANGE RATE GENERAL	2	CHRATG	16	CRG
CHANGE SS NAME	2	CHSSN	13	CSN
CHANGE SS PREC	2	CHSSPR	23	CSP
CHANGE SS RATE	2	CHSSRT	18	CSR
CHANGE SS VALUE	3	CHSSV	68	
CHANGE UNITS ENTRY	2	CHUNIE	24	CUE
CHANGE UNITS GENERAL	2	CHUNIG	25	CUG
CHECK CONSISTENCY.	2	CKCONS	38	CCS
CHECK CORRELATION	2	CKCORR	39	CCR
CHECK REFERENCE	4	CKREF	101	
CHECK SUBSET	4	CKSUB	107	
CHECK TA	3	CKTA	81	
CHECK TA SET	3	CKTAST	88	
CLEAR CF LISTS	4	CLRCFL	94	
CLEAR DATABASE	2	CLRDB	31	CLR
CLEAR MESSAGE LISTS	4	CLRMEL	95	
CLISTREF	6	CLISTR	122	
CLOSE LIST	3	CLOLIS	87	
COMMAND WARN	3	CMDWRN	57	
CONFIGURE SYSTEM	2	CONFIG	32	CFS
CONVERT NAME	4	CONVNM	98	
DELETE DATA REFERENCE	2	DELDAR	27	DDR
DELETE ENTRY	3	DELENT	69	
DELETE NAME	5	DELNAM	115	
DELETE ONE ENTRY	2	DEL1EN	28	DOE
DISMANTLE SYSTEM	2	DISMAN	51	DMS
FATAL CHECK	3	FATLCK	74	
FATAL PRINT	4	FATLPR	105	
FILE CW	4	FILCW	109	
FILE DIRECTS	4	FILDIR	111	
FILE ENTRIES	2	FILENT	29	FEN
FILE HEADER	5	FILHDR	118	
FILE MESSAGES	2	FILMES	50	FLM
FILE NAMELIST	3	FILNAM	83	
FILE NUMBER	6	FILNUM	123	
FILE ONE SA	4	FIL1SA	110	
FILE REFLIST	3	FILREF	70	
FILE SCHEDULES	2	FILSCH	54	FLS
FILE SUBADDRESSES	3	FILSAD	91	
FILE SUBSYSTEM DATA	2	FILESD	30	FSD
FILE TABLES	3	FILTAB	90	
FIND BUS SS	3	FINBSS	76	
FIND BUSCON	4	FINBC	108	
FIND CODE	3	FINCOD	84	

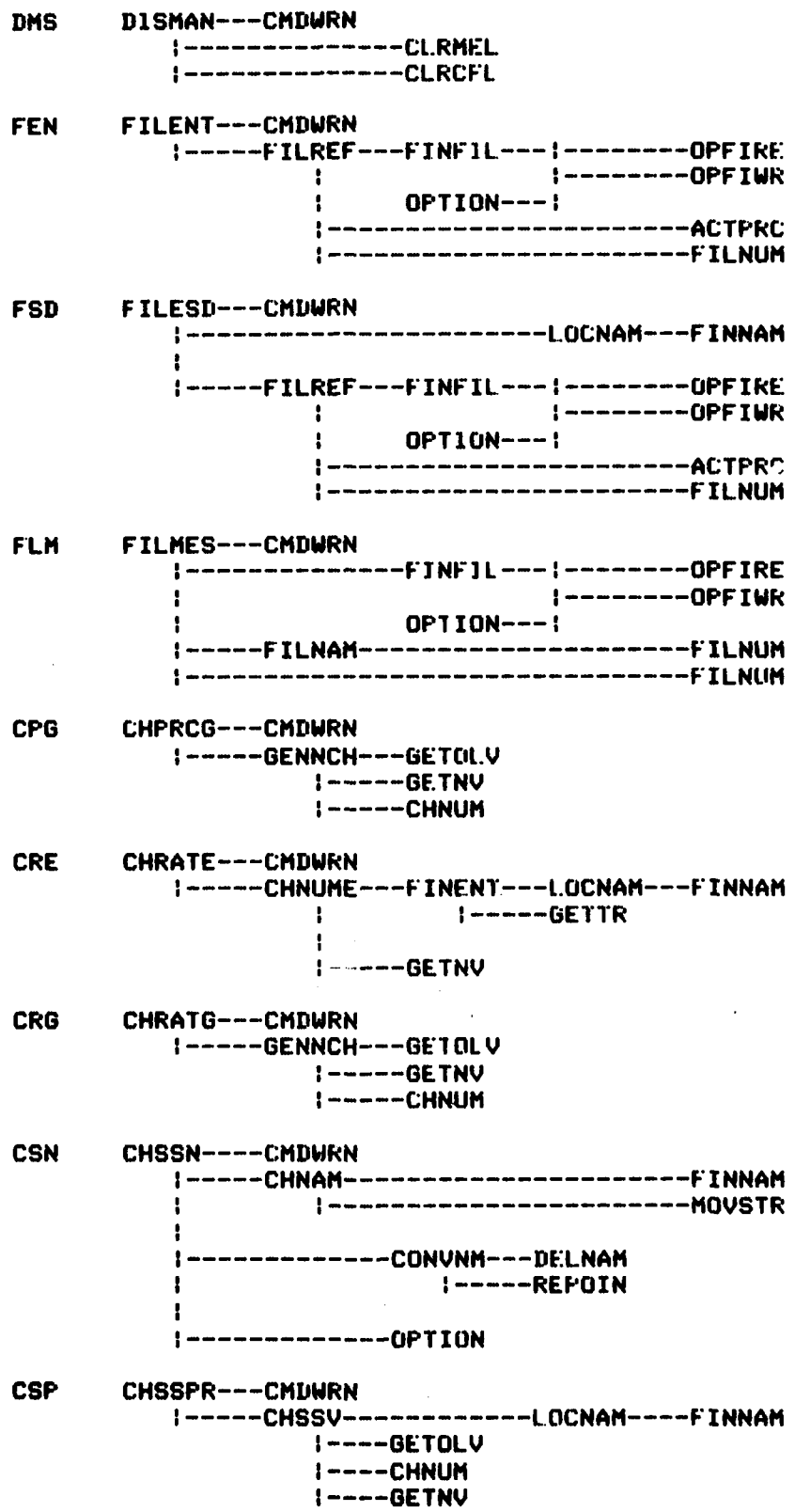
FIND ENTRY	4	FINENT	100	
FIND FILE	4	FINFIL	93	
FIND MESSAGE	3	FINMES	80	
FIND NAME	6	FINNAM	119	
FLO PARAM CHANGE	3	FLOPCH	59	
FORM MESSAGES	3	FRMMES	75	
FORM RX SS	3	FRMRXS	73	
FORM SUBADDRESSES	3	FRMSAD	89	
FORM SUBSETS	3	FRMSUB	85	
FORM SYSTEM	2	FRMSYS	52	FMS
FORM TX DATA	3	FRMTXD	72	
FORM TX SS	3	FRMTXS	71	
GENERAL NUMBER CHANGE	3	GENNCH	66	
GET NEW VALUE	4	GETNV	99	
GET OLD VALUE	4	GETOLV	102	
GET TR	5	GETTR	113	
INITIALISE	3	INIT	56	
INPUT DATA	3	INDAT	60	
INPUT FILED ENTRIES	2	INFILE	6	IFE
INPUT FILED MESSAGES	2	INFILM	7	IFM
INPUT NAMELIST	3	INNAM	61	
INPUT TERMINAL ENTRIES	2	INTTYE	5	ITE
INT PARAM CHANGE	3	INTPCH	58	
LIST BUS MESSAGES	2	LISMES	40	LBM
LIST COMMANDS	4	LISCMD	92	
LIST CONFIGURED SUBSYSTEMS	2	LISCOS	34	LCS
LIST DATA NAMES	2	LISDNM	11	LDN
LIST DATA PATHS	2	LISDAP	33	LDP
LIST DATA TRAFFIC	2	LISDAT	35	LDT
LIST DIRECT DATA	2	LISDDA	41	LDD
LIST ENTRIES	2	LISENT	8	LEN
LIST MESSAGE DATA	3	LISMDA	82	
LIST NAMES	3	LISNAM	63	
LIST ONE MESSAGE	2	LIS1ME	44	LOM
LIST REFLIST	3	LISREF	62	
LIST SS DATA	2	LISSSD	9	LSD
LIST SS NAMES	2	LISSSN	10	LSN
LIST SS PAIR	3	LISSSP	79	
LIST SUBADDRESSES	2	LISSAD	53	LSA
LIST SUBSETS	2	LISSUB	47	LSS
LOCATE NAME	5	LOCNAM	114	
MAIN PROGRAM	0	SAV2MN	1	
MAP DATA TRAFFIC	2	MAPDAT	36	MDT
MOVE STRING	6	MOVSTR	120	
OBEY COMMAND	1	OBEYCM	3	
OPEN LIST	3	OPELIS	86	
OPTION	4	OPTION	96	
READ ENTRY	4	READEN	97	
RECORD NAME	5	RECNAM	112	
REFORMAT MESSAGES	2	REFMTM	49	RFM
REORDER MESSAGE	2	REORDM	48	ROM
REPOINT	5	REPOIN	116	
RESET PARAMETERS	2	RESET	4	RSP
SEND CONTENTS	5	SENCON	117	
SEND MESSAGE	4	SENMES	106	
SET COMMAND STRINGS	1	SETCMD	2	
SET MAP	3	SETMAP	78	
SET RETRY CODE	2	SETRET	45	SRC
SET TERMINAL ADDRESS	2	SETTAD	46	STA
PARTITION NUMBER	3	PARNUM	77	
SOURCE SINK PAIR DATA	2	SSPRDA	37	SSP
SUMMARISE MESSAGES	2	SUMMES	43	SUM

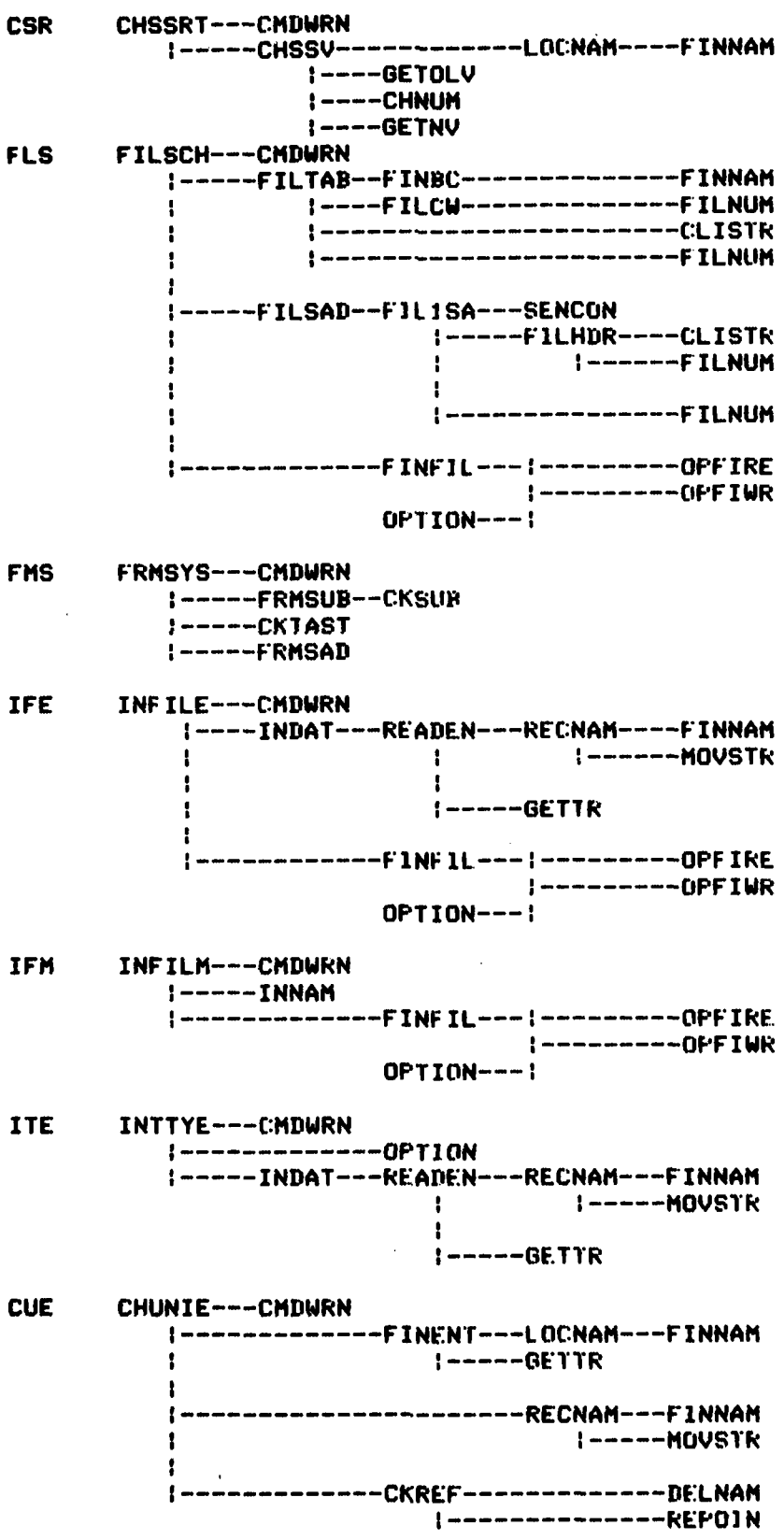
TRACE DATA PATH	2	TRDATA	12	TDP
UNFORM SYSTEM	2	UNFSYS	55	UFS
WORDS	4	WORDS	104	

APPENDIX B

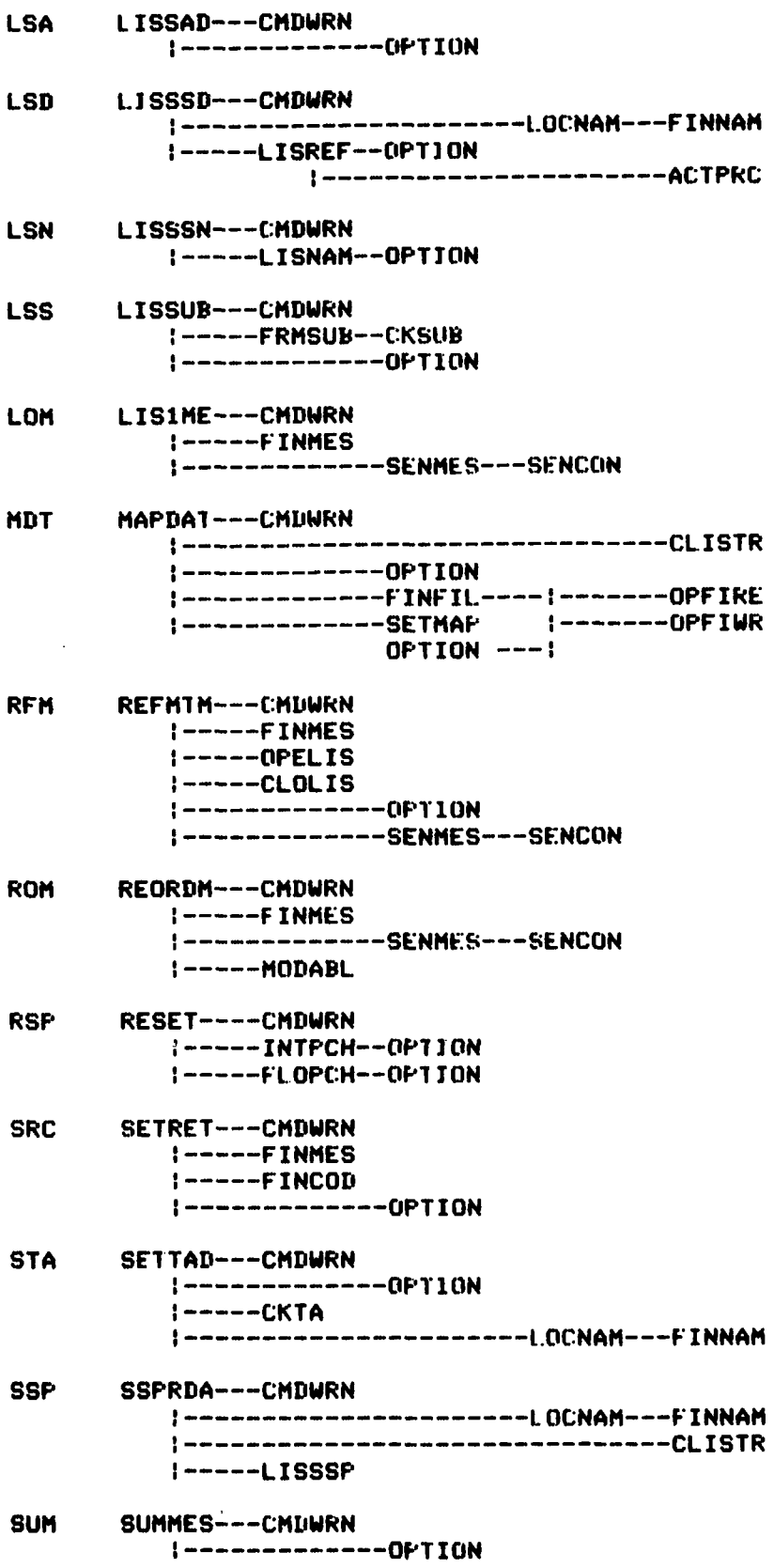
PROGRAM STRUCTURE




DMS
DISMAN---CMDWRN
CLRMEL
CLRCFL
FEN
FILENT---CMDWRN
FILREF---FINFIL---OPFIRE
OPFIWR
OPTION---
ACTPRC
FILNUM
FSD
FILESD---CMDWRN
LOCNAM---FINNAM
FILREF---FINFIL---OPFIRE
OPFIWR
OPTION---
ACTPRC
FILNUM
FLM
FILMES---CMDWRN
FINFIL---OPFIRE
OPFIWR
OPTION---
FILNAM---FILNUM
FILNUM
CPG
CHPRCG---CMDWRN
GENNCH---GETOLV
GETNV
CHNUM
CRE
CHRATE---CMDWRN
CHNUME---FINENT---LOCNAM---FINNAM
GETTR
GETNV
CRG
CHRATG---CMDWRN
GENNCH---GETOLV
GETNV
CHNUM
CSN
CHSSN----CMDWRN
CHNAM---FINNAM
MOVSTR
CONVNM---DELNAM
REPOIN
OPTION
CSP
CHSSPR---CMDWRN
CHSSV---LOCNAM----FINNAM
GETOLV
CHNUM
GETNV


CSR
CHSSRT---CMDWRN
-----CHSSV----------LOCNAM----FINNAM
----GETOLV
----CHNUM
----GETNV
FLS
FILSCH---CMDWRN
-----FILTAB--FINBC----------FINNAM
----FILCW----------FILNUM
----------CLISTR
----------FILNUM
-----FILSAD--FILISA---SENCON
-----FILHDR----CLISTR
------FILNUM
----------FILNUM
----------FINFIL---|---------OPFIRE
|---------OPFIWR
OPTION---|
FMS
FRMSYS---CMDWRN
-----FRMSUB--CKSUB
-----CKTAST
-----FRMSAD
IFE
INFILE---CMDWRN
----INDAT---READEN---RECNAM----FINNAM
------MOVSTR
-----GETTR
----------FINFIL---|---------OPFIRE
|---------OPFIWR
OPTION---|
IFM
INFILM---CMDWRN
-----INNAM
----------FINFIL---|---------OPFIRE
|---------OPFIWR
OPTION---|
ITE
INTTYE---CMDWRN
----------OPTION
-----INDAT---READEN---RECNAM---FINNAM
-----MOVSTR
-----GETTR
CUE
CHUNIE---CMDWRN
----------FINENT---LOCNAM---FINNAM
-----GETTR
----------RECNAM---FINNAM
-----MOVSTR
----------CKREF----------DELNAM
----------REPOIN

```
CUG     CHUNIG---CMDWRN
           |-----CHNAM----------------------FINNAM
           |          |----------------------MOVSTR
           |
           |-------------OPTION
           |-------------CONVNM---DELNAM
                               |-----REPOIN

CCS     CKCONS---CMDWRN
           |-----FATLCK--FATLPR------------ACTPRC
                       |---------------------ACTPRC

CCR     CKCORR---CMDWRN

CLR     CLRDB----CMDWRN
           |-----INIT----CLRMEL
           |          |----CLRCFL
           |          |----LISCMD
           |
           |-------------OPTION

CFS     CONFIG---CMDWRN
           |-----FRMTXD
           |-----FRMRXS--WORDS
           |          |---------------------ACTPRC
           |
           |-----FATLCK--FATLPR
           |          |---------------------ACTPRC
           |
           |-----FRMMES
           |-----FINBSS
           |-----FRMTXS
           |-----PARNUM

LDP     LISDAP---CMDWRN
           |-------------OPTION
           |-----------------------------ACTPRC

LDT     LISDAT---CMDWRN
           |-------------OPTION
           |-----LISSSP

LDD     LISDDA---CMDWRN
           |-----LISMDA
                      |----OPTION
                      |----SENMES---SENCON

LDN     LISDNM---CMDWRN
           |-----LISNAM--OPTION

LEN     LISENT---CMDWRN
           |-----LISREF--OPTION
                      |---------------------ACTPRC

LBM     LISMES---CMDWRN
           |-----LISMDA--OPTION
           |-------------SENMES---SENCON
```


LSA
LISSAD---CMDWRN
OPTION
LSD
LISSSD---CMDWRN
LOCNAM---FINNAM
LISREF--OPTION
ACTPRC
LSN
LISSSN---CMDWRN
LISNAM--OPTION
LSS
LISSUB---CMDWRN
FRMSUB--CKSUB
OPTION
LOM
LIS1ME---CMDWRN
FINMES
SENMES---SENCON
MDT
MAPDAT---CMDWRN
CLISTR
OPTION
FINFIL
OPFIRE
SETMAP
OPFIWR
OPTION
RFM
REFMTM---CMDWRN
FINMES
OPELIS
CLOLIS
OPTION
SENMES---SENCON
ROM
REORDM---CMDWRN
FINMES
SENMES---SENCON
MODABL
RSP
RESET----CMDWRN
INTPCH--OPTION
FLOPCH--OPTION
SRC
SETRET---CMDWRN
FINMES
FINCOD
OPTION
STA
SETTAD---CMDWRN
OPTION
CKTA
LOCNAM---FINNAM
SSP
SSPRDA---CMDWRN
LOCNAM---FINNAM
CLISTR
LISSSP
SUM
SUMMES---CMDWRN
OPTION

```
TDP     TRDATA---CMDWRN
           ¦----------------------LOCNAM---FINNAM
           ¦-------------------------------ACTPRC

UFS     UNFSYS---CMDWRN
```

APPENDIX C

TYPICAL LIB/OVL FILES

```
'CORAL' NUM00
'INCLUDE' "SY:CORIOL.CRL"
'COMMON' ('LABEL' OVL00;
          'INCLUDE' "SY:PROCS.COR"
                                    );
'INCLUDE' "SY:SAVGBL.COR"
'SEGMENT' OVL00
'BEGIN'
        'INCLUDE' "SY:MACDEF.COR"
        'INCLUDE' "SY:CALL0A.COR"
'END'
'FINISH'


'CORAL' MOD00
'INCLUDE' "SY:CORIOL.CRL"
'COMMON' ('LABEL' LIB00;
          'INCLUDE' "SY:PROCS.COR"
                                    );
'INCLUDE' "SY:SAVGBL.COR"
'SEGMENT' LIB00
'BEGIN'
        'INCLUDE' "SY:MACDEF.COR"
        'INCLUDE' "SY:ACTPRC.COR"
'END'
'FINISH'
```

APPENDIX D

OVL/LIB SEGMENT NAME CORRESPONDENCE

LIB/MOD	SEGMENT NAME	OVL/NUM	SEGMENT NAME
LIB00	ACTPRC	OVL00	CALLOA
LIB01	CHDAV	OVL01	CHDAPR
LIB02	CHNAM	OVL02	CHDART
LIB03	CHNUM	OVL03	CHDAUN
LIB04	CHNUME	OVL04	CHDNM
LIB05	CHSSV	OVL05	CHPRCE
LIB06	CKREF	OVL06	CHPRCG
LIB07	CKSUB	OVL07	CHRATE
LIB08	CKTA	OVL08	CHRATG
LIB09	CKTAST	OVL09	CHSSN
LIB10	CLISTR	OVL10	CHSSPR
LIB11	CLOLIS	OVL11	CHSSRT
LIB12	CLRCFL	OVL12	CHUNIE
LIB13	CLRMEL	OVL13	CHUNIG
LIB14	CONVNM	OVL14	CKCONS
LIB15	DELENT	OVL15	CKCORR
LIB16	DELNAM	OVL16	CLRDB
LIB17	FATLCK	OVL17	CONFIG
LIB18	FATLPR	OVL18	DEL1EN
LIB19	FILCW	OVL19	DELDAR
LIB20	FILDIR	OVL20	DISMAN
LIB21	FILHDR	OVL21	FILENT
LIB22	FILNAM	OVL22	FILESD
LIB23	FILNUM	OVL23	FILMES
LIB24	FILREF	OVL24	FILSCH
LIB25	FILSAD	OVL25	FRMSYS
LIB26	FILTAB	OVL26	INFILE
LIB27	FIL1SA	OVL27	INFILM
LIB28	FINBC	OVL28	INIT
LIB29	FINBSS	OVL29	INTTYE
LIB30	FINCOD	OVL30	LISCOS
LIB31	FINENT	OVL31	LISDAP
LIB32	FINFIL	OVL32	LISDAT
LIB33	FINMES	OVL33	LISDDA
LIB34	FINNAM	OVL34	LISDNM
LIB35	FLOPCH	OVL35	LISENT
LIB36	FRMMES	OVL36	LISMES
LIB37	FRMRXS	OVL37	LISSAD
LIB38	FRMSAD	OVL38	LISSSD
LIB39	FRMSUB	OVL39	LISSSN
LIB40	FRMTXD	OVL40	LISSUB
LIB41	FRMTXS	OVL41	LIS1ME
LIB42	GENNCH	OVL42	MAPDAT
LIB43	GETNV	OVL43	REFMTM
LIB44	GETOLV	OVL44	REORDM
LIB45	GETTR	OVL45	RESET
LIB46	INDAT	OVL46	SETRET
LIB47	INNAM	OVL47	SETTAD
LIB48	INTPCH	OVL48	SSPRDA
LIB49	LISCMD	OVL49	SUMMES
LIB50	LISMDA	OVL50	TRDATA
LIB51	LISNAM	OVL51	UNFSYS
LIB52	LISREF		

LIB53	LISSSP
LIB54	LOCNAM
LIB55	MOVSTR
LIB56	OPELIS
LIB57	READEN
LIB58	OPFIRE
LIB59	RECNAM
LIB60	REPOIN
LIB61	SENCON
LIB62	SENMES
LIB63	SETMAP
LIB64	PARNUM
LIB65	WORDS
LIB66	OPFIWR
LIB67	MODABL

APPENDIX E1

GLOBAL FILES

```
'NOLIST'

'COMMENT' CORIOL.CRL - DEC CORAL LIBRARY SPECS;

'COMMENT' MACRO DEFINITIONS FOR LIBRARY SPECS;

'DEFINE' VB "'VALUE''BYTE'";
'DEFINE' VI "'VALUE''INTEGER'";
'DEFINE' VF "'VALUE''FLOATING'";
'DEFINE' LB "'LOCATION''BYTE'";
'DEFINE' LI "'LOCATION''INTEGER'";
'DEFINE' LF "'LOCATION''FLOATING'";
'DEFINE' BA "'BYTE''ARRAY'";

'COMMENT' LIBRARY ROUTINES FROM COROTS.OLB;

'LIBRARY' ( 'PROCEDURE' NEWLINE (VB,VB);
            'PROCEDURE' DEFLUN (VB,VI,'LABEL');
            'PROCEDURE' DELETE (VB);
            'PROCEDURE' CREATE (VB,VI,VI,VB,'LABEL');
            'PROCEDURE' OPEN (VB,VI,VI,VB,'LABEL');
            'PROCEDURE' WRITETEXT (VB,VI);
            'PROCEDURE' READCH (VB,LB);
            'PROCEDURE' SPACES (VB,VB);
            'PROCEDURE' IWRITE (VB,VI,VI);
            'PROCEDURE' FWRITE (VB,VF,VI);
            'PROCEDURE' WRITECH (VB,VB);
            'PROCEDURE' IREAD (VB,VB,LI);
            'PROCEDURE' FREAD (VB,LF);
            'PROCEDURE' TIDEF (VB,VB,'LABEL');
            'PROCEDURE' CLOSE (VB);
            'PROCEDURE' RELEASE (VB);
            'PROCEDURE' READTEXT (VB,VI,BA,LI);
            'PROCEDURE' WRITEREC (VB,BA,VI,LI);
            'PROCEDURE' ERRSET (VI,VI,VI,VI,VI,BA)
          );

'LIST'
'NOLIST'
'COMMENT'     PROCS.COR
              -------- ;

   'INTEGER' 'PROCEDURE' ACTPRC ('VALUE''INTEGER') ;
             'PROCEDURE' CALLOA ;
             'PROCEDURE' CHDAPR ;
             'PROCEDURE' CHDART ;
             'PROCEDURE' CHDAUN ;
             'PROCEDURE' CHDAV ('VALUE''INTEGER') ;
             'PROCEDURE' CHDSAV ('VALUE''INTEGER') ;
             'PROCEDURE' CHDNM ;
             'PROCEDURE' CHNAM ('BYTE''ARRAY') ;
             'PROCEDURE' CHNUM ('VALUE''INTEGER','VALUE''INTEGER','VALUE''INTEGER','VALUE''INTEGER') ;
             'PROCEDURE' CHNUME ('VALUE''INTEGER') ;
             'PROCEDURE' CHPRCE ;
             'PROCEDURE' CHPRCG ;
```

```
            'PROCEDURE' CHRATE ;
            'PROCEDURE' CHRATG ;
            'PROCEDURE' CHSSN ;
            'PROCEDURE' CHSSPR ;
            'PROCEDURE' CHSSRT ;
            'PROCEDURE' CHSSV ('VALUE''INTEGER') ;
            'PROCEDURE' CHUNIE ;
            'PROCEDURE' CHUNIG ;
            'PROCEDURE' CKCONS ;
            'PROCEDURE' CKCORR ;
            'PROCEDURE' CKREF ('BYTE''ARRAY','VALUE''INTEGER') ;
            'PROCEDURE' CKSUB ('VALUE''INTEGER','VALUE''INTEGER','LOCATION''INTEGER') ;
            'PROCEDURE' CKTA ('VALUE''INTEGER','VALUE''INTEGER','VALUE''INTEGER','LOCATION''INTEGER');
            'PROCEDURE' CKTAST ('LOCATION''INTEGER') ;
'INTEGER'   'PROCEDURE' CLISTR ('VALUE''INTEGER') ;
            'PROCEDURE' CLOLIS ('VALUE''INTEGER') ;
            'PROCEDURE' CLRCFL ;
            'PROCEDURE' CLRDB ;
            'PROCEDURE' CLRMEL ;
            'PROCEDURE' CMDWRN ('VALUE''INTEGER') ;
'INTEGER'   'PROCEDURE' CMPSTR ('VALUE''INTEGER','VALUE''INTEGER') ;
            'PROCEDURE' CONFIG ;
            'PROCEDURE' CONVNM ('BYTE''ARRAY','VALUE''INTEGER','VALUE''INTEGER') ;
            'PROCEDURE' DELDAR ;
            'PROCEDURE' DELENT ('VALUE''INTEGER') ;
            'PROCEDURE' DELNAM ('BYTE''ARRAY','VALUE''INTEGER') ;
            'PROCEDURE' DEL1EN ;
            'PROCEDURE' DISMAN ;
            'PROCEDURE' FATLCK ('LOCATION''INTEGER') ;
            'PROCEDURE' FATLPR ('VALUE''INTEGER','VALUE''INTEGER','VALUE''INTEGER','LOCATION''INTEGER') ;
            'PROCEDURE' FILCW ('VALUE''INTEGER','VALUE''INTEGER','VALUE''INTEGER','VALUE''INTEGER') ;
            'PROCEDURE' FILDIR ('VALUE''INTEGER','VALUE''INTEGER','LOCATION''INTEGER');
            'PROCEDURE' FILENT ;
            'PROCEDURE' FILESD ;
            'PROCEDURE' FILHDR ('VALUE''INTEGER') ;
            'PROCEDURE' FILMES ;
            'PROCEDURE' FILNAM ('BYTE''ARRAY') ;
            'PROCEDURE' FILNUM ('VALUE''INTEGER') ;
            'PROCEDURE' FILREF ('VALUE''INTEGER') ;
            'PROCEDURE' FILSAD ;
            'PROCEDURE' FILSCH ;
            'PROCEDURE' FILTAB ;
            'PROCEDURE' FIL1SA ('VALUE''INTEGER','VALUE''INTEGER','VALUE''INTEGER','VALUE''INTEGER','LOCATION''INTEGER');
            'PROCEDURE' FINBC ('LOCATION''INTEGER') ;
            'PROCEDURE' FINBSS ('LOCATION''INTEGER') ;
            'PROCEDURE' FINCOD ('LOCATION''INTEGER') ;
            'PROCEDURE' FINENT ('LOCATION''INTEGER','LOCATION''INTEGER','LOCATION''INTEGER') ;
            'PROCEDURE' FINFIL ('VALUE''INTEGER','VALUE''INTEGER','LOCATION' 'INTEGER') ;
            'PROCEDURE' FINMES ('VALUE''INTEGER','LOCATION''INTEGER') ;
            'PROCEDURE' FINNAM ('BYTE''ARRAY','LOCATION''INTEGER') ;
            'PROCEDURE' FLOPCH ('VALUE''INTEGER','VALUE''FLOATING','LOCATION''FLOATING');
            'PROCEDURE' FRMMES ;
            'PROCEDURE' FRMRXS ;
            'PROCEDURE' FRMSAD ('LOCATION''INTEGER') ;
            'PROCEDURE' FRMSUB ;
            'PROCEDURE' FRMSYS ;
            'PROCEDURE' FRMTXD ;
            'PROCEDURE' FRMTXS ;
            'PROCEDURE' GENNCH ('VALUE''INTEGER') ;
'FLOATING'  'PROCEDURE' GETNUM ('VALUE''BYTE','BYTE''ARRAY') ;
            'PROCEDURE' GETNV ('VALUE''INTEGER','LOCATION''INTEGER') ;
            'PROCEDURE' GETOLV ('VALUE''INTEGER','LOCATION''INTEGER') ;
```

```
                'PROCEDURE' GETTR ('VALUE''BYTE','LOCATION''INTEGER') ;
                'PROCEDURE' INDAT ('VALUE''BYTE','VALUE''INTEGER','LOCATION''INTEGER','LOCATION''INTEGER');
                'PROCEDURE' INFILE ;
                'PROCEDURE' INFILM ;
                'PROCEDURE' INIT ;
                'PROCEDURE' INNAM ('BYTE''ARRAY') ;
                'PROCEDURE' INTPCH ('VALUE''INTEGER','VALUE''INTEGER','VALUE''INTEGER','VALUE''INTEGER','LOCATION''INTEGER');
                'PROCEDURE' INTTYE ;
                'PROCEDURE' LISCMD ;
                'PROCEDURE' LISCOS ;
                'PROCEDURE' LISDAP ;
                'PROCEDURE' LISDAT ;
                'PROCEDURE' LISDDA ;
                'PROCEDURE' LISDNM ;
                'PROCEDURE' LISENT ;
                'PROCEDURE' LISMDA ('VALUE''INTEGER') ;
                'PROCEDURE' LISMES ;
                'PROCEDURE' LISNAM ('BYTE''ARRAY','VALUE''INTEGER') ;
                'PROCEDURE' LISREF ('VALUE''INTEGER') ;
                'PROCEDURE' LISSAD ;
                'PROCEDURE' LISSSD ;
                'PROCEDURE' LISSSN ;
                'PROCEDURE' LISSSP ('VALUE''INTEGER','VALUE''INTEGER','LOCATION''INTEGER') ;
                'PROCEDURE' LISSUB ;
                'PROCEDURE' LISTME ;
                'PROCEDURE' LOCNAM ('VALUE''INTEGER','BYTE''ARRAY','LOCATION''INTEGER') ;
                'PROCEDURE' MAPDAT ;
                'PROCEDURE' MODABL ('VALUE''INTEGER','VALUE''INTEGER','VALUE''INTEGER','VALUE''INTEGER','VALUE''INTEGER');
                'PROCEDURE' MOVSTR ('BYTE''ARRAY','VALUE''INTEGER') ;
                'PROCEDURE' OBEYCM ;
                'PROCEDURE' OPELIS ('VALUE''INTEGER') ;
   'INTEGER'    'PROCEDURE' OPTION ('VALUE''INTEGER') ;
                'PROCEDURE' PARNUM;
                'PROCEDURE' READEN ('VALUE''BYTE','VALUE''INTEGER','LOCATION''INTEGER','LOCATION''INTEGER');
     'INTEGER''PROCEDURE' OPFIRE;
     'INTEGER''PROCEDURE' OPFIWR;
                'PROCEDURE' RECNAM ('BYTE''ARRAY','VALUE''INTEGER','LOCATION''INTEGER','LOCATION''INTEGER') ;
                'PROCEDURE' REFMTM ;
                'PROCEDURE' REORDM ;
                'PROCEDURE' REPOIN ('VALUE''INTEGER','VALUE''INTEGER') ;
                'PROCEDURE' RESET ;
                'PROCEDURE' SENCON ('VALUE''INTEGER','VALUE''INTEGER','VALUE''INTEGER') ;
                'PROCEDURE' SENMES ('VALUE''INTEGER') ;
                'PROCEDURE' SETCMD ;
                'PROCEDURE' SETMAP ;
                'PROCEDURE' SETRET ;
                'PROCEDURE' SETTAD ;
                'PROCEDURE' SLNO ;
                'PROCEDURE' SSPRDA ;
                'PROCEDURE' SUMMES ;
                'PROCEDURE' TRDATA ;
                'PROCEDURE' UNFSYS ;
   'INTEGER'    'PROCEDURE' WORDS ('VALUE''INTEGER') ;
'LIST'
'NOLIST'
   'COMMENT'    ---------SAVGBL.COR-----------------;


   'COMMENT'    ----DEFINITIONS AND DECLARATIONS----;


   'COMMENT'    PROGRAM CONSTANTS;
```

```
'DEFINE' MAXENTRIES '500'; (*** Max number of database entries allowed)
'DEFINE' MAXMESS '200'; (*** Max number of data bus messages allowed)
'DEFINE' MAXSS '50'; (*** Max number of subsystem names)
'DEFINE' MAXDATA '120'; (*** Max number of data names)
'DEFINE' MAXSTWDS '13'; (*** Max number of words to hold a STRING - gives 13 char. STRING)
'DEFINE' TTO '1'; (*** Channel number for terminal input)
'DEFINE' TTI '2'; (*** Channel number for terminal output)
'DEFINE' INFIL '3'; (*** Channel number for disc file input access)
'DEFINE' OUTFIL '4'; (*** Channel number for disc file output access)


'COMMENT' PROGRAM STATES.
***   These macros define the various values taken
***   by the variable STATE;

'DEFINE' STOPPED '0';
'DEFINE' EMPTY '1';
'DEFINE' OPENX '2';
'DEFINE' CONFIGURED '3';
'DEFINE' LIMITED '4';
'DEFINE' FORMED '5';
'DEFINE' LIMITED FORMED '6';
'DEFINE' INCONSISTENT '7';


'COMMENT' PROGRAMMING AIDS.
***   These macros are designed to make coding more
***   explicit and to aid structured programming;

'DEFINE' WHILE (COND) ''FOR' WHLOOP:=0 'WHILE' COND';
   (*** Gives contrived while-loop clause)
'DEFINE' TRUE '1'; (*** For pseudo boolean operations)
'DEFINE' FALSE '0'; (*** Ditto)
'DEFINE' NOT(X) ''IF' X=TRUE 'THEN' FALSE 'ELSE' TRUE';  (*** Ditto)
'DEFINE' INC(X) 'X:=X+1';
'DEFINE' DEC(X) 'X:=X-1';
'DEFINE' DO NOTHING '';
'DEFINE' STRING(LOC) ''LOCATION'(LOC[0])'; (*** Points to start of string in 'LOC' BUFFER)



'COMMON' GLOBAL (
   'COMMENT' INTEGER CONSTANTS.
   ***   These are the preset integers defining the particular
   ***   system architecture, whose values can be changed by
   ***   commanding RSP;

   'INTEGER' LOWEST RATE, WORD LENGTH, MESSAGE LENGTH, WORD OVERHEAD, BIT RATE, TERMINAL LIMIT
      :=8, 16, 32, 4, 1000, 30;

   'COMMENT' REAL CONSTANTS.
   ***   Similarly, these are the real preset constants
   ***   whose values can be changed by commanding RSP;

   'FLOATING' CONT OVERHEAD, TERM OVERHEAD := 2.6, 4.9;

   'COMMENT' GLOBAL INTEGERS;

   'INTEGER' ENTRIES, TOTAL TX DATA, TOTAL MESSAGES, CONF SS COUNT, STATE, WHLOOP,OBEYED;
```

```
'COMMENT'
###   ENTRIES: Count of REFLIST entries.
###   TOTAL TX DATA: Count of TXLIST entries.
###   TOTAL MESSAGES: Count of MESSAGE HEADER & MESSAGE DATA entries.
###   CONF SS COUNT: Count of configured subsystems.
###   STATE: The current database state.
###   WHLOOP: Dummy variable for contrived while-loop;

'COMMENT' INTEGERS FOR FREQUENTLY USED STRINGS.
###   These hold pointers to text strings which appear frequently
###   within the program.;

'INTEGER' EACH PAGE, CONTINUE;

'COMMENT'
###   EACH PAGE: "TO PAUSE ON EACH PAGE".
###   CONTINUE: "TO CONTINUE";

'COMMENT' TEMPORARY VARIABLES;
'BYTE''ARRAY' ATEMP[0:19],ERRSTAT[0:2];
'INTEGER' ITEMP,JTEMP;

'COMMENT' DATABASE ARRAYS;

'BYTE''ARRAY' SS NAME, UNITS NAME[0:MAXSS,0:MAXSTWDS],DATA NAME[0:MAXDATA,0:MAXSTWDS];
'INTEGER''ARRAY' CONF SS LIST, TERM ADDRESS, SUBADD COUNT[1:MAXSS], MESSAGE DATA[0:MAXMESS,0:32], MAP[-1:MAXSS,-1:MAXSS];

'COMMENT'
###   SS NAME: Namelist array for subsystem name strings.
###   DATA NAME: Namelist array for data item name strings.
###   UNITS NAME: Namelist array for units name strings.
###   CONF SS LIST: Array for list of configured subsystem codes.
###   TERM ADDRESS: Array for terminal address values of subsystems in
###                 CONF SS LIST.  Value of 0 means subsystem on bus
###                 but ta not set. Value of -1 means ss not on bus.
###   SUBADD COUNT: Array for counts of subaddress assignments made
###                 for s ›ssystems in CONF SS LIST.
###   MESSAGE DATA: Array to hold codes of data items in bus messages,
###                 each ¬ow representing one message.
###   MAP: Array used for formulation of source/sink pair map;

'COMMENT' TEXT BUFFERS;

'BYTE''ARRAY' STRBUFF, FNAME[0:MAXSTWDS];
'BYTE''ARRAY' REMARKS[0:41], INBUFF, OUTBUFF[0:101];

'COMMENT'
###   STRBUFF: Buffer for all text strings typed by user except file
###            names and remarks for files.
###   FNAME: Buffer for all name text.
###   REMARKS: Buffer for file remarks text.
###   INBUFF: Character buffer required by library input procedures.
###   OUTBUFF: ditto for output procedures;


'COMMENT' ARRAY FOR COMMAND LISTING STRINGS;
'INTEGER''ARRAY' CMD[0:52];

'COMMENT' PRESET ARRAY DEFINING COMMANDS IN EACH STATE;
'INTEGER''ARRAY' CMDLIST[1:7,0:40]:=
7,1,2,0,3,4,0,52,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,
40,5,6,0,7,8,0,9,10,0,11,12,0,13,14,0,15,16,0,17,18,0,19,20,0,21,22,0,23,24,0,2,3,0,25,26,0,27,28,0,52,
```

```
38,5,6,0,7,8,0,29,30,0,31,32,0,33,35,0,36,37,0,38,39,0,40,41,0,42,43,0,44,45,0,25,26,0,46,47,0,48,52,,,
26,7,8,0,30,32,0,36,37,0,38,39,0,40,41,0,42,43,0,44,45,0,46,48,0,27,52,,,,,,,,,,,,,,,
35,5,6,0,7,8,0,9,29,0,30,31,0,32,33,0,35,36,0,37,38,0,39,40,0,41,25,0,26,46,0,49,50,0,51,52,,,,,,
22,7,8,0,30,32,0,36,37,0,38,39,0,40,41,0,46,49,0,50,51,0,52,,,,,,,,,,,,,,,,,,
19,5,6,0,7,8,0,9,29,0,30,34,0,35,25,0,26,47,0,52;


'COMMENT'
***    CMD: Set up by SET COMMAND STRINGS with pointers to all
***         the text strings used when displaying available commands.
***         CMD[0] is set to a string which just gives a new line, for
***         convenience when interpreting the CMDLIST array.
***    CMDLIST: The procedure used to list the commands available
***             in the current state is LIST COMMANDS. It does
***             this with reference to the preset CMDLIST array, where
***             each row represents the state whose value is the row
***             number (ex: EMPTY - 1, OPEN - 2, etc) and references
***             the commands valid in that state.  This array is
***             preset with pointers to the CMD array entries which
***             point to the actual strings.  The first number in
***             each row is the number of following entries in
***             the row, and then the commands are grouped in twos,
***             separated by 0's, which are pointers to the new line
***             strings.  Thus the commands are listed two to a line
***             on the display.  For example, Row 1, which represents
***             the EMPTY state has 7 entries: 1 (RSP), 2 (ITE),
***             0 (newline), 3 (IFE), 4 (IFM), 0 (newline) and
***             52 (STOP);

'COMMENT' DATABASE TABLES.
***    REFLIST holds the list of basic database entries, each defining
***    one data input or output requirement for a subsystem.
***    TXLIST is used when the system is configured to hold the list
***    of all transmitted data items, with their associated transmitting
***    subsystems, rate, precision and units identifiers.  RXLIST is
***    used to hold information on receivers of the transmitted data
***    held in TXLIST.  For each transmitted data item RXLIST holds the
***    receiving subsystem codes together with their rate, precision
***    and units requirements.  MESSAGE HEADER holds the specification
***    of each bus message (or direct data transfer) in the configured
***    system, the contents of which are held in the corresponding rows
***    of the MESSAGE DATA array.
***    This is a convenient point to say that since table entries
***    are referenced from index 0, the counts of entries on the various
***    tables (ENTRIES, TOTAL TX DATA and TOTAL MESSAGES) always point
***    to the next available entry.  Search loops, then are always
***    indexed from 0 to 1 less than the count value - ex:
***    'FOR' REFPTR := 0 'STEP' 1 'UNTIL' ENTRIES -1 'DO'...
***    SAVANT always increments message numbers by 1 when interfacing with
***    the user, so that the message which the user knows as MESSAGE 1,
***    say, is actually held in entry 0 of the table.  In comments
***    within the program, we talk about the 'user message number'
***    and the 'actual message number', and this is the difference;

'TABLE' REFLIST[3,MAXENTRIES]
[SS 'UNSIGNED' (6) 0'BIT'0;
DATA 'UNSIGNED' (9) 0'BIT'6;
TR 'UNSIGNED' (1) 0'BIT'15;
RATE 'UNSIGNED' (4) 1'BIT'0;
PREC 'UNSIGNED' (10) 1'BIT'4;
CONF 'UNSIGNED' (1) 1'BIT'14;
UNITS 'UNSIGNED' (6) 2'BIT'0];
```

```
'COMMENT'
***   SS: The subsystem name code in the entry.
***   DATA: The data name code.
***   TR: The transmit/receive bit (1: transmit, 0: receive).
***   RATE: The rate group number.
***   PREC: Precision in bits.
***   CONF: Flag to indicate entry dealt with during configuration.
***   UNITS: The units name code;

'TABLE' TXLIST[4,MAXDATA]
[TXDATA 'UNSIGNED' (9) 0'BIT'0;
TXSS 'UNSIGNED' (6) 0'BIT'9;
TXRATE 'UNSIGNED' (4) 1'BIT'0;
TXPREC 'UNSIGNED' (10) 1'BIT'4;
TXUNITS 'UNSIGNED' (6) 2'BIT'0;
TXCHECK 'UNSIGNED' (1) 2'BIT'6;
NRX 'UNSIGNED' (6) 3'BIT'0;
RXSTART 'UNSIGNED' (10) 3'BIT'6];

'COMMENT'
***    TXDATA: The transmitted data item.
***   TXSS: Its transmitting subsystem.
***   TXRATE: Its rate.
***   TXPREC: Its precision.
***   TXUNITS: Its units.
***   TXCHECK: Flag used when checking consistency.
***   NRX: The number of receivers for the data.
***   RXSTART: Pointer to the first RXLIST entry for this data;

'TABLE' RXLIST[2,MAXENTRIES]
[RXSS 'UNSIGNED' (6) 0'BIT'0;
RXRATE 'UNSIGNED' (4) 0'BIT'6;
RXUNITS 'UNSIGNED' (6) 0'BIT'10;
RXPREC 'UNSIGNED' (10) 1'BIT'0;
RXWORDS 'UNSIGNED' (6) 1'BIT'10];

'COMMENT'
***   RXSS: The receiving subsystem.
***   RXRATE: Its rate requirement.
***   RXPREC: Its precision requirement.
***   RXUNITS: Its units requirement.
***   RXWORDS: Number of words represented by RXPREC;

'TABLE' MESSAGE HEADER[3,MAXMESS]
[TX 'UNSIGNED' (6) 0'BIT'0;
RX 'UNSIGNED' (6) 0'BIT'6;
RG 'UNSIGNED' (4) 0'BIT'12;
WDS 'UNSIGNED' (6) 1'BIT'0;
WC 'UNSIGNED' (5) 1'BIT'0;
TXSA 'UNSIGNED' (5) 1'BIT'6;
RXSA 'UNSIGNED' (5) 1'BIT'11;
RETRY 'UNSIGNED' (4) 2'BIT'0;
SUBSET 'UNSIGNED' (8) 2'BIT'4;
EXCEPT 'UNSIGNED' (1) 2'BIT'12]

'COMMENT'
***   TX: The source subsystem.
***   RX: The sink subsystem.
```

```
   ***   RG: The rate group for the message.
   ***   WDS: Number of data words in message.
   ***   WC: Subfield of WDS for bus schedule.
   ***   TXSA: The subaddress of the message in the source.
   ***   RXSA: The subaddress of the message in the sink.
   ***   RETRY: The entry code for the message.
   ***   SUBSET: Pointer to a message of which this message is a subset.
   ***           The user message number is used for the pointer, allowing
   ***           a value of 0 to mean no subset.
   ***   EXCEPT: Flag used when setting retry codes;
                                                                    );



'LIST'
'NOLIST'
   'COMMENT' --------------MACDEF.COR-------------------;

   'COMMENT'  FILE HANDLING MACROS;

   'DEFINE' OPEN READ FILE (L1,L2)"DEFLUN(INFIL,'LOCATION'(FNAME[0]),L1);OPEN(INFIL,""S"",""RP"",1,L2)";
   'DEFINE' OPEN WRITE FILE (L1,L2)"DEFLUN(OUTFIL,'LOCATION'(FNAME[0]),L1);CREATE(OUTFIL,""S"",0,1,L2)";
   'DEFINE' CLOSE READ FILE "CLOSE(INFIL);RELEASE(INFIL)";
   'DEFINE' CLOSE WRITE FILE "CLOSE(OUTFIL);RELEASE(OUTFIL)";
   'DEFINE' FILENAME VALID "TRUE";
   'DEFINE' FILE EXISTS "TRUE";


   'COMMENT'  INPUT MACROS.
    ***  Macros defining calls on input library procedures.
    ***  'DEV' is the device, either 'TTI' or 'INFIL';

   'DEFINE' READ NUM(DEV) "GETNUM(DEV,INBUFF)";
      (*** Reads a number from device - assumed type procedure)
   'DEFINE' INPUT TEXT(DEV) "READTEXT(DEV,0,STRBUFF,ITEMP)";
      (*** Reads a text string of up to 13 characters from 'DEV' into STRBUFF)
   'DEFINE' READ FILE NAME "READTEXT(TTI,0,FNAME,ITEMP)";
      (*** Reads a text string of up to 7 characters from 'TTI' into FNAME)
   'DEFINE' READ ONE(CHAR)
   "'BEGIN'
        READTEXT(TTI,0,ATEMP,ITEMP);
        CHAR:=ATEMP[2]
   'END'";
      (*** Reads one character from 'TTI' into CHAR)
   'DEFINE' READ REMARKS(DEV) "READTEXT(DEV,0,REMARKS,ITEMP)";
      (*** Reads a text string of up to 80 characters from 'DEV' into REMARKS)


   'COMMENT'  OUTPUT MACROS.
   ***   Macros defining calls on output library procedures.
   ***   'DEV' is device, either 'TTO' or 'OUTFIL';

   'DEFINE' WRITE NUM(DEV,NUM,D1,D2) "IWRITE(DEV,(NUM),""I6"")";
      (*** Writes 'NUM' to device where D1 & D2 are layout parameters)
   'DEFINE' SEND CHAR(DEV,CH) "WRITECH(DEV,(CH))";
      (*** Sends one character to device)
   'DEFINE' OUTPUT SPC(DEV,N) "SPACES(DEV,N)";
      (*** Writes 'N' spaces to device)
   'DEFINE' ENDLINE(DEV,N) "NEWLINE(DEV,N)";
      (*** Writes 'N' new lines to device)
```

```
'DEFINE' OUTPUT TEXT(DEV,TEXT) 'WRITETEXT(DEV,TEXT)';
   (*** Writes text string to device)
'DEFINE' NAME(DEV,NL,POS) 'WRITETEXT(DEV,'LOCATION'(NL[POS,0]))';
   (*** Writes the name whose code is 'POS' from the namelist 'NL' to device)
'DEFINE' NAME TAB(DEV,NL,POS,NS)
   'SPACES(DEV,NS-NL[POS,0])';
   (*** Follows up output of name with spaces to give a total
    *** field width of NS spaces, for tabulation purposes)


'COMMENT'  STRING COMPARISON MACROS.
***   Note: STRBUFF is where all input strings, apart from file
***   names and file remarks, are buffered prior to comparison;

'DEFINE' TEXT IS(STR) 'CMPSTR('LOCATION'(STRBUFF[0]),STR)=0';
   (*** Delivers TRUE value if string in 'STRBUFF' = string 'STR')
'DEFINE' NAME MATCHES(NL,POS)
   'CMPSTR('LOCATION'(STRBUFF[0]),'LOCATION'(NL[POS,0]))=0';
   (*** TRUE if string in 'STRBUFF' = name in namelist 'NL' at index 'POS')


'COMMENT'  PARTWORD MACROS.
***   To access the two different part word fields in
***   the words of the message data array;

'DEFINE' PARTITION COUNT(M,W) ''BITS'[7,9]MESSAGE DATA[M,W]';
'DEFINE' DATAWORD(M,W) ''BITS'[9,0]MESSAGE DATA[M,W]';
'LIST'
```

APPENDIX E2

ROOT SEGMENT

```
'CORAL' $MAIN
'INCLUDE' 'LB:[1,1]CORIOL.CRL'
'COMMON' ('LABEL' ROOT;
          'INCLUDE' 'LB:[32,1]PROCS.COR'
                                  );
'INCLUDE' 'LB:[32,1]SAVGBL.COR'
'ENTER' ROOT;
'SEGMENT' ROOT
'BEGIN'
      'INCLUDE' 'LB:[32,1]MACDEF.COR'
   'INTEGER''PROCEDURE' OPTION ('VALUE''INTEGER' PROMPT);
   'BEGIN'

      'COMMENT'                                                                    OPTION (LEVEL 4)
      $$$   Used whenever the user needs to respond to a question with
      $$$   a simple yes/no answer.  The question is posed - the nature
      $$$   of the question being delivered as the PROMPT string - and
      $$$   OPTION takes the value TRUE only if 'Y' is typed in response.
      $$$   Called by: INTTYE       LISDNM
      $$$              CLRDB        LISDAP
      $$$              LISDAT       MAPDAT
      $$$              SUMMES       SETRET
      $$$              SETTAD       LISSUB
      $$$              REFMTM       LISSAD
      $$$              INTPCH       FLOPCH
      $$$              LISREF       LISNAM
      $$$              CHNAM        FRMTXS
      $$$              LISMDA       FINFIL;

      'INTEGER' REPLY;

      'COMMENT'
      $$$   REPLY: The user's reply;

      ENDLINE(TTO,1);
      OUTPUT TEXT(TTO,'TYPE 'Y' ');
      OUTPUT TEXT(TTO,PROMPT); OUTPUT TEXT(TTO,' .... ');
      READ ONE(REPLY);
      ENDLINE(TTO,1);
      OUTPUT TEXT(TTO,' .... OK');
      ENDLINE(TTO,1);
      'ANSWER'('IF' REPLY='LITERAL'(Y) 'THEN' TRUE 'ELSE' FALSE)
   'END' OPTION;


   'PROCEDURE' CMDWRN ('VALUE''INTEGER' TYPE);
   'BEGIN'

      'COMMENT'                                                                    CMDWRN (LEVEL 3)
      $$$   Warns of invalid command and reminds of listing option.
      $$$   If TYPE is delivered as 0, the command is completely
      $$$   unknown, whereas if TYPE is 1 then the command has been
      $$$   recognised but the user has attempted to invoke it in
      $$$   a state in which it is not valid.  The erroneous command is
      $$$   repeated back to the user in the warning message.
```

```
      ***   Called by: OBEY COMMAND and all Level 2 procedures;

      ENDLINE(TTO,1);
      OUTPUT TEXT(TTO,'COMMAND ');
      OUTPUT TEXT(TTO,STRING(STRBUFF));
      'IF' TYPE=0 'THEN'
'BEGIN' OUTPUT TEXT(TTO,' NOT UNDERSTOOD');
      ENDLINE(TTO,1)
      'END'
      'ELSE'
      'BEGIN'
       OUTPUT TEXT(TTO,' NOT VALID IN THIS STATE');
       ENDLINE(TTO,1)
      'END';
        ENDLINE(TTO,1);
        OUTPUT TEXT(TTO,'TYPE 'H' TO LIST VALID COMMANDS');
        ENDLINE(TTO,1);
   'END' CMDWRN;


'FLOATING''PROCEDURE' GETNUM('VALUE''BYTE' DEV;'BYTE''ARRAY' INBUFF);
'BEGIN'
        'FLOATING' NUMBER;
        FREAD(DEV,NUMBER);
        'ANSWER' NUMBER
'END' GETNUM;

'INTEGER''PROCEDURE' CMPSTR('VALUE''INTEGER'STRING1,STRING2);
'BEGIN'
        'COMMENT' Compares STRING1 and STRING2 character by character.
        Returns a value of:     +1      STRING1 longer than STRING2
                                -1      STRING2 longer than STRING1
                                +2      STRING1 and STRING2 are different
                                 0      STRING1 and STRING2 are identical;
        'BYTE''ARRAY' CHAR[1:2];
        'INTEGER' LENGTH1,LENGTH2,NEXT CHAR;
        LENGTH1:=[STRING1];
        LENGTH2:=[STRING2];
        'IF' LENGTH1>LENGTH2 'THEN' 'ANSWER' +1;
        'IF' LENGTH1<LENGTH2 'THEN' 'ANSWER' -1;
        'FOR' NEXT CHAR:=1 'STEP' 1 'UNTIL' LENGTH1 'DO'
        'BEGIN'
                CHAR[1]:=[STRING1+NEXT CHAR+1];
                CHAR[2]:=[STRING2+NEXT CHAR+1];
                'IF' CHAR[1]<>CHAR[2] 'THEN' 'ANSWER' +2
        'END';
        'ANSWER' 0
'END' CMPSTR;
   'PROCEDURE' SETCMD;
   'BEGIN'

      'COMMENT'                                                          SETCMD (LEVEL 1)
      ***   Sets up pointers to the command strings in the CMD array, for
      ***   use when listing commands.  Also sets up two other frequently
      ***   used strings.
      ***   Called by: SAVD21;

      CMD[0]:="
";
      CMD[1]:="RSP - RESET PARAMETERS              ";
      CMD[2]:="ITE - INPUT TERMINAL ENTRIES        ";
      CMD[3]:="IFE - INPUT FILED ENTRIES           ";
```

```
   CMD[4]:='IFM - INPUT FILED MESSAGES          ';
   CMD[5]:='LEN - LIST ENTRIES                  ';
   CMD[6]:='LSD - LIST SUBSYSTEM DATA           ';
   CMD[7]:='LSN - LIST SUBSYSTEM NAMES          ';
   CMD[8]:='LDN - LIST DATA NAMES               ';
   CMD[9]:='TDP - TRACE DATA PATH               ';
   CMD[10]:='CSN - CHANGE SUBSYSTEM NAME         ';
   CMD[11]:='CDN - CHANGE DATA NAME              ';
   CMD[12]:='CRE - CHANGE RATE ENTRY             ';
   CMD[13]:='CRG - CHANGE RATE GENERAL           ';
   CMD[14]:='CDR - CHANGE DATA RATE              ';
   CMD[15]:='CSR - CHANGE SUBSYSTEM RATE         ';
   CMD[16]:='CPE - CHANGE PREC ENTRY             ';
   CMD[17]:='CPG - CHANGE PREC GENERAL           ';
   CMD[18]:='CDP - CHANGE DATA PREC              ';
   CMD[19]:='CSP - CHANGE SUBSYSTEM PREC         ';
   CMD[20]:='CUE - CHANGE UNITS ENTRY            ';
   CMD[21]:='CUG - CHANGE UNITS GENERAL          ';
   CMD[22]:='CDU - CHANGE DATA UNITS             ';
   CMD[23]:='DDR - DELETE DATA REFERENCE         ';
   CMD[24]:='DOE - DELETE ONE ENTRY              ';
   CMD[25]:='FEN - FILE ENTRIES                  ';
   CMD[26]:='FSD - FILE SUBSYSTEM DATA           ';
   CMD[27]:='CLR - CLEAR DATABASE                ';
   CMD[28]:='CFS - CONFIGURE SYSTEM              ';
   CMD[29]:='LDP - LIST DATA PATHS               ';
   CMD[30]:='LCS - LIST CONFIGURED SUBSYSTEMS    ';
   CMD[31]:='LDT - LIST DATA TRAFFIC             ';
   CMD[32]:='MDT - MAP DATA TRAFFIC              ';
   CMD[33]:='SSP - SOURCE SINK PAIR DATA         ';
   CMD[34]:='CCS - CHECK CONSISTENCY             ';
   CMD[35]:='CCR - CHECK CORRELATION             ';
   CMD[36]:='SUM - SUMMARISE MESSAGES            ';
   CMD[37]:='LBM - LIST BUS MESSAGES             ';
   CMD[38]:='LDD - LIST DIRECT DATA              ';
   CMD[39]:='CLD - CALCULATE LOADINGS            ';
   CMD[40]:='LOM - LIST ONE MESSAGE              ';
   CMD[41]:='LSS - LIST SUBSETS                  ';
   CMD[42]:='SRC - SET RETRY CODE                ';
   CMD[43]:='STA - SET TERMINAL ADDRESS          ';
   CMD[44]:='ROM - REORDER MESSAGE               ';
   CMD[45]:='RFM - REFORMAT MESSAGES             ';
   CMD[46]:='FLM - FILE MESSAGES                 ';
   CMD[47]:='DMS - DISMANTLE SYSTEM              ';
   CMD[48]:='FMS - FORM SYSTEM                   ';
   CMD[49]:='LSA - LIST SUBADDRESSES             ';
   CMD[50]:='FLS - FILE SCHEDULES                ';
   CMD[51]:='UFS - UNFORM SYSTEM                 ';
   CMD[52]:='STOP - RETURN TO RSX-11/M           ';
   EACH PAGE:='TO PAUSE ON EACH PAGE';
   CONTINUE:='TO CONTINUE, ANYTHING ELSE TO STOP'
'END' SETCMD;


              'PROCEDURE' OBEYCM;
'BEGIN'

   'COMMENT'                                              OBEYCM - OBEY  COMMAND (LEVEL 1)
   ***   The command interpreter.
   ***   Calls the appropriate Level 2 procedure depending on the
   ***   input command text string. If the command is not recognised
   ***   then a notification is sent.
```

```
  ***   Called by: MAIN PROGRAM
  ***   Calls       CMDWRN          LISCMD
  ***               All Level 2 procedures;

  OBEYED:=TRUE ;

  'IF' TEXT IS('RSP') 'THEN' RESET
  'ELSE''IF' TEXT IS('ITE') 'THEN' INTTYE
  'ELSE''IF' TEXT IS('IFE') 'THEN' INFILE
  'ELSE''IF' TEXT IS('IFM') 'THEN' INFILM
  'ELSE''IF' TEXT IS('LEN') 'THEN' LISENT
  'ELSE''IF' TEXT IS('LSD') 'THEN' LISSSD
  'ELSE''IF' TEXT IS('LSN') 'THEN' LISSSN
  'ELSE''IF' TEXT IS('LDN') 'THEN' LISDNM
  'ELSE''IF' TEXT IS('TDP') 'THEN' TRDATA
  'ELSE''IF' TEXT IS('CSN') 'THEN' CHSSN
  'ELSE''IF' TEXT IS('CDN') 'THEN' CHDNM
  'ELSE''IF' TEXT IS('CRE') 'THEN' CHRATE
  'ELSE''IF' TEXT IS('CRG') 'THEN' CHRATG
  'ELSE''IF' TEXT IS('CDR') 'THEN' CHDART
  'ELSE''IF' TEXT IS('CSR') 'THEN' CHSSRT
  'ELSE''IF' TEXT IS('CPE') 'THEN' CHPRCE
  'ELSE''IF' TEXT IS('CPG') 'THEN' CHPRCG
  'ELSE' OBEYED:=FALSE;
  'IF' OBEYED = FALSE 'THEN'
'BEGIN'
  OBEYED:= TRUE;
  'IF' TEXT IS('CDP') 'THEN' CHDAPR
  'ELSE''IF' TEXT IS('CSP') 'THEN' CHSSPR
  'ELSE''IF' TEXT IS('CUE') 'THEN' CHUNIE
  'ELSE''IF' TEXT IS('CUG') 'THEN' CHUNIG
  'ELSE''IF' TEXT IS('CDU') 'THEN' CHDAUN
  'ELSE''IF' TEXT IS('DDR') 'THEN' DELDAR
  'ELSE''IF' TEXT IS('DOE') 'THEN' DEL1EN
  'ELSE''IF' TEXT IS('FEN') 'THEN' FILENT
  'ELSE''IF' TEXT IS('FSD') 'THEN' FILESD
  'ELSE''IF' TEXT IS('CLR') 'THEN' CLRDB
  'ELSE''IF' TEXT IS('CFS') 'THEN' CONFIG
  'ELSE''IF' TEXT IS('LDP') 'THEN' LISDAP
  'ELSE''IF' TEXT IS('LCS') 'THEN' LISCOS
  'ELSE''IF' TEXT IS('LDT') 'THEN' LISDAT
  'ELSE''IF' TEXT IS('MDT') 'THEN' MAPDAT
  'ELSE''IF' TEXT IS('SSP') 'THEN' SSPRDA
  'ELSE''IF' TEXT IS('CCS') 'THEN' CKCONS
  'ELSE''IF' TEXT IS('CCR') 'THEN' CKCORR
  'ELSE' OBEYED:=FALSE ;
'END';
  'IF' OBEYED = FALSE 'THEN'
'BEGIN'
  'IF' TEXT IS('LBM') 'THEN' LISMES
  'ELSE''IF' TEXT IS('LDD') 'THEN' LISDDA
  'ELSE''IF' TEXT IS('CLD') 'THEN' CALLOA
  'ELSE''IF' TEXT IS('SUM') 'THEN' SUMMES
  'ELSE''IF' TEXT IS('LOM') 'THEN' LIS1ME
  'ELSE''IF' TEXT IS('LSS') 'THEN' LISSUB
  'ELSE''IF' TEXT IS('SRC') 'THEN' SETRET
  'ELSE''IF' TEXT IS('STA') 'THEN' SETTAD
  'ELSE''IF' TEXT IS('ROM') 'THEN' REORDM
  'ELSE''IF' TEXT IS('RFM') 'THEN' REFMTM
  'ELSE''IF' TEXT IS('FLM') 'THEN' FILMES
  'ELSE''IF' TEXT IS('DMS') 'THEN' DISMAN
  'ELSE''IF' TEXT IS('FMS') 'THEN' FRMSYS
```

```
      'ELSE''IF' TEXT IS('LSA') 'THEN' LISSAD
      'ELSE''IF' TEXT IS('FLS') 'THEN' FILSCH
      'ELSE''IF' TEXT IS('UFS') 'THEN' UNFSYS
      'ELSE''IF' TEXT IS('H') 'THEN' LISCMD
      'ELSE'
      'BEGIN'
         'COMMENT' *** command not recognised;
         CMDWRN(0)
      'END'
   'END'
'END' OBEYCM;


   'PROCEDURE' INIT;
   'BEGIN'

      'COMMENT'                                                                    INIT (LEVEL 3)
      ***   Returns the database to the EMPTY state, initialising counts
      ***   and all database tables and arrays.
      ***   Called by: SAVD21          CLRDB
      ***   Calls:     LISCMD          CLRCFL
      ***              CLRMEL;

      'INTEGER' PTR1, PTR2;

      'COMMENT'
      ***   PTR1: Pointer used as index for clearing REFLIST and namelists.
      ***   PTR2: Pointer used as index to individual namelist words;

      CLRMEL;
      CLRCFL;
      'FOR' PTR1 := 0 'STEP' 1 'UNTIL' 3*MAXENTRIES - 1 'DO' REFLIST[PTR1] := 0;
      ENTRIES := 0;
      'FOR' PTR2 := 0 'STEP' 1 'UNTIL' 6 'DO'
      'BEGIN'
         'FOR' PTR1 := 0 'STEP' 1 'UNTIL' MAXSS 'DO'
         'BEGIN'
            SS NAME[PTR1, PTR2] := 0;
            UNITS NAME[PTR1, PTR2] := 0
         'END';
         'FOR' PTR1 := 0 'STEP' 1 'UNTIL' MAXDATA 'DO' DATA NAME[PTR1, PTR2] := 0
      'END';
      SS NAME[0, 1] := MAXSS;
      SS NAME[0, 2] := 0;
      DATA NAME[0, 1] := MAXDATA;
      DATA NAME[0, 2] := 1;
      UNITS NAME[0, 1] := MAXSS;
      UNITS NAME[0, 2] := 2;
      STATE := EMPTY;
      LISCMD
   'END' INIT;
   'COMMENT' System Architecture Verification & Analysis Technique

   SAVANT 2 - Version 5/1 of 26 JUL 81

   Originally written by A A Callaway FS5 Div RAE Farnborough Hants

   for a PRIME 300.

   This version of SAVANT has been modified by R Bluff and

   P T May to run on a PDP 11/34 ;
```

```
   TIDEF(TTO,TTI,TIERR);
   ERRSET(0,0,-1,0,0,ERRSTAT);
   ENDLINE(TTO,1);
   OUTPUT TEXT(TTO,'WELCOME TO SAVANT 2 - VERSION: 5/1 OF 26 JUL 81');
   ENDLINE(TTO,1);
   SETCMD;
   INIT;
   WHILE (STATE<>STOPPED) 'DO'
   'BEGIN'
      ENDLINE(TTO,2);
      OUTPUT TEXT(TTO,'COMMAND .... ');
      INPUT TEXT(TTI);
      ENDLINE(TTO,1);
      'IF' TEXT IS('STOP') 'THEN'
           'BEGIN'
           ENDLINE(TTO,1);
           OUTPUT TEXT(TTO,'HAVE YOU FILED ALL REQUIRED DATA ?');
           'IF' OPTION('IF YOU REALLY WANT TO STOP')=TRUE
                'THEN' STATE:=STUPPED
           'END'
           'ELSE' OBEYCM
   'END';
TIERR:


'END'
'FINISH'
```

APPENDIX E3

LIST OF USER PROCEDURES

```
'COMMENT'
 ***   Please note !!! This file will not compile and is only a list
 ***   of all the SAVANT procedures for reference purposes;


'PROCEDURE' SETCMD;
'BEGIN'

   'COMMENT'                                                          SETCMD (LEVEL 1)
   ***   Sets up pointers to the command strings in the CMD array, for
   ***   use when listing commands.  Also sets up two other frequently
   ***   used strings.
   ***   Called by: SAVD21;

   CMD[0]:="

   CMD[1]:="RSP - RESET PARAMETERS              ";
   CMD[2]:="ITE - INPUT TERMINAL ENTRIES        ";
   CMD[3]:="IFE - INPUT FILED ENTRIES           ";
   CMD[4]:="IFM - INPUT FILED MESSAGES          ";
   CMD[5]:="LEN - LIST ENTRIES                  ";
   CMD[6]:="LSD - LIST SUBSYSTEM DATA           ";
   CMD[7]:="LSN - LIST SUBSYSTEM NAMES          ";
   CMD[8]:="LDN - LIST DATA NAMES               ";
   CMD[9]:="TDP - TRACE DATA PATH               ";
   CMD[10]:="CSN - CHANGE SUBSYSTEM NAME         ";
   CMD[11]:="CDN - CHANGE DATA NAME              ";
   CMD[12]:="CRE - CHANGE RATE ENTRY             ";
   CMD[13]:="CRG - CHANGE RATE GENERAL           ";
   CMD[14]:="CDR - CHANGE DATA RATE              ";
   CMD[15]:="CSR - CHANGE SUBSYSTEM RATE         ";
   CMD[16]:="CPE - CHANGE PREC ENTRY             ";
   CMD[17]:="CPG - CHANGE PREC GENERAL           ";
   CMD[18]:="CDP - CHANGE DATA PREC              ";
   CMD[19]:="CSP - CHANGE SUBSYSTEM PREC         ";
   CMD[20]:="CUE - CHANGE UNITS ENTRY            ";
   CMD[21]:="CUG - CHANGE UNITS GENERAL          ";
   CMD[22]:="CDU - CHANGE DATA UNITS             ";
   CMD[23]:="DDR - DELETE DATA REFERENCE         ";
   CMD[24]:="DOE - DELETE ONE ENTRY              ";
   CMD[25]:="FEN - FILE ENTRIES                  ";
   CMD[26]:="FSD - FILE SUBSYSTEM DATA           ";
   CMD[27]:="CLR - CLEAR DATABASE                ";
   CMD[28]:="CFS - CONFIGURE SYSTEM              ";
   CMD[29]:="LDP - LIST DATA PATHS               ";
   CMD[30]:="LCS - LIST CONFIGURED SUBSYSTEMS    ";
   CMD[31]:="LDT - LIST DATA TRAFFIC             ";
   CMD[32]:="MDT - MAP DATA TRAFFIC              ";
   CMD[33]:="SSP - SOURCE SINK PAIR DATA         ";
   CMD[34]:="CCS - CHECK CONSISTENCY             ";
   CMD[35]:="CCR - CHECK CORRELATION             ";
   CMD[36]:="SUM - SUMMARISE MESSAGES            ";
   CMD[37]:="LBM - LIST BUS MESSAGES             ";
   CMD[38]:="LDD - LIST DIRECT DATA              ";
   CMD[39]:="CLD - CALCULATE LOADINGS            ";
   CMD[40]:="LOM - LIST ONE MESSAGE              ";
   CMD[41]:="LSS - LIST SUBSETS                  ";
```

```
   CMD[42]:='SRC - SET RETRY CODE                    ';
   CMD[43]:='STA - SET TERMINAL ADDRESS              ';
   CMD[44]:='ROM - REORDER MESSAGE                   ';
   CMD[45]:='RFM - REFORMAT MESSAGES                 ';
   CMD[46]:='FLM - FILE MESSAGES                     ';
   CMD[47]:='DMS - DISMANTLE SYSTEM                  ';
   CMD[48]:='FMS - FORM SYSTEM                       ';
   CMD[49]:='LSA - LIST SUBADDRESSES                 ';
   CMD[50]:='FLS - FILE SCHEDULES                    ';
   CMD[51]:='UFS - UNFORM SYSTEM                     ';
   CMD[52]:='STOP - RETURN TO RSX-11/M               ';
   EACH PAGE:='TO PAUSE ON EACH PAGE';
   CONTINUE:='TO CONTINUE, ANYTHING ELSE TO STOP'
'END' SETCMD;


              'PROCEDURE' OBEYCM;
'BEGIN'

   'COMMENT'                                                    OBEYCM - OBEY  COMMAND (LEVEL 1)
   ***   The command interpreter.
   ***   Calls the appropriate Level 2 procedure depending on the
   ***   input command text string. If the command is not recognised
   ***   then a notification is sent.
   ***   Called by: MAIN PROGRAM
   ***   Calls       CMDWRN          LISCMD
   ***               All Level 2 procedures;

   OBEYED:=TRUE ;

   'IF' TEXT IS('RSP') 'THEN' RESET
   'ELSE''IF' TEXT IS('ITE') 'THEN' INTTYE
   'ELSE''IF' TEXT IS('IFE') 'THEN' INFILE
   'ELSE''IF' TEXT IS('IFM') 'THEN' INFILM
   'ELSE''IF' TEXT IS('LEN') 'THEN' LISENT
   'ELSE''IF' TEXT IS('LSD') 'THEN' LISSSD
   'ELSE''IF' TEXT IS('LSN') 'THEN' LISSSN
   'ELSE''IF' TEXT IS('LDN') 'THEN' LISDNM
   'ELSE''IF' TEXT IS('TDP') 'THEN' TRDATA
   'ELSE''IF' TEXT IS('CSN') 'THEN' CHSSN
   'ELSE''IF' TEXT IS('CDN') 'THEN' CHDNM
   'ELSE''IF' TEXT IS('CRE') 'THEN' CHRATE
   'ELSE''IF' TEXT IS('CRG') 'THEN' CHRATG
   'ELSE''IF' TEXT IS('CDR') 'THEN' CHDART
   'ELSE''IF' TEXT IS('CSR') 'THEN' CHSSRT
   'ELSE''IF' TEXT IS('CPE') 'THEN' CHPRCE
   'ELSE''IF' TEXT IS('CPG') 'THEN' CHPRCG
   'ELSE' OBEYED:=FALSE;
   'IF' OBEYED = FALSE 'THEN'
 'BEGIN'
   OBEYED:= TRUE;
   'IF' TEXT IS('CDP') 'THEN' CHDAPR
   'ELSE''IF' TEXT IS('CSP') 'THEN' CHSSPR
   'ELSE''IF' TEXT IS('CUE') 'THEN' CHUNIE
   'ELSE''IF' TEXT IS('CUG') 'THEN' CHUNIG
   'ELSE''IF' TEXT IS('CDU') 'THEN' CHDAUN
   'ELSE''IF' TEXT IS('DDR') 'THEN' DELDAR
   'ELSE''IF' TEXT IS('DOE') 'THEN' DEL1EN
   'ELSE''IF' TEXT IS('FEN') 'THEN' FILENT
   'ELSE''IF' TEXT IS('FSD') 'THEN' FILESD
   'ELSE''IF' TEXT IS('CLR') 'THEN' CLRDB
   'ELSE''IF' TEXT IS('CFS') 'THEN' CONFIG
```

```
        'ELSE''IF' TEXT IS('LDP') 'THEN' LISDAP
        'ELSE''IF' TEXT IS('LCS') 'THEN' LISCOS
        'ELSE''IF' TEXT IS('LDT') 'THEN' LISDAT
        'ELSE''IF' TEXT IS('MDT') 'THEN' MAPDAT
        'ELSE''IF' TEXT IS('SSP') 'THEN' SSPRDA
        'ELSE''IF' TEXT IS('CCS') 'THEN' CKCONS
        'ELSE''IF' TEXT IS('CCR') 'THEN' CKCORR
        'ELSE' OBEYED:=FALSE ;
      'END';
        'IF' OBEYED = FALSE 'THEN'
      'BEGIN'
        'IF' TEXT IS('LBM') 'THEN' LISMES
        'ELSE''IF' TEXT IS('LDD') 'THEN' LISDDA
        'ELSE''IF' TEXT IS('CLD') 'THEN' CALLOA
        'ELSE''IF' TEXT IS('SUM') 'THEN' SUMMES
        'ELSE''IF' TEXT IS('LOM') 'THEN' LIS1ME
        'ELSE''IF' TEXT IS('LSS') 'THEN' LISSUB
        'ELSE''IF' TEXT IS('SRC') 'THEN' SETRET
        'ELSE''IF' TEXT IS('STA') 'THEN' SETTAD
        'ELSE''IF' TEXT IS('ROM') 'THEN' REORDM
        'ELSE''IF' TEXT IS('RFM') 'THEN' REFMTM
        'ELSE''IF' TEXT IS('FLM') 'THEN' FILMES
        'ELSE''IF' TEXT IS('DMS') 'THEN' DISMAN
        'ELSE''IF' TEXT IS('FMS') 'THEN' FRMSYS
        'ELSE''IF' TEXT IS('LSA') 'THEN' LISSAD
        'ELSE''IF' TEXT IS('FLS') 'THEN' FILSCH
        'ELSE''IF' TEXT IS('UFS') 'THEN' UNFSYS
        'ELSE''IF' TEXT IS('H') 'THEN' LISCMD
        'ELSE'
        'BEGIN'
          'COMMENT' *** command not recognised;
          CMDWRN(0)
        'END'
      'END'
'END' OBEYCM;


  'PROCEDURE' RESET;
  'BEGIN'

    'COMMENT'                                                                    RESET (LEVEL 2)
    ***   SAVANT Command.
    ***   This procedure allows the preset system parameters to be reset.
    ***   Each is offered in turn, and the new value entered if IVAL
    ***   or RVAL are returned non-negative.
    ***   Called by: OBEY COMMAND
    ***   Calls:     CMDWRN            INTPCH
    ***              FLOPCH;

    'IF' STATE <> EMPTY
    'THEN' CMDWRN (1)
    'ELSE'
    'BEGIN'
      'INTEGER' IVAL;
      'FLOATING' FVAL;

      'COMMENT'
      ***   IVAL: Integer value returned from change procedure.
      ***   FVAL: Floating value returned from change procedure;

      INTPCH ('WORD LENGTH', 'BITS', WORD LENGTH, 0, IVAL);
      'IF' IVAL >= 0 'THEN' WORD LENGTH := IVAL;
```

```
      INTPCH ('MESSAGE LENGTH', 'WORDS', MESSAGE LENGTH, 63, IVAL);
      'IF' IVAL >= 0 'THEN' MESSAGE LENGTH := IVAL;
      INTPCH ('NUMBER OF RATE GROUPS', '', LOWEST RATE, 15, IVAL);
      'IF' IVAL >= 0 'THEN' LOWEST RATE := IVAL;
      INTPCH ('WORD OVERHEAD (SYNC + PARITY)', 'BITS', WORD OVERHEAD, 0, IVAL);
      'IF' IVAL >= 0 'THEN' WORD OVERHEAD := IVAL;
      INTPCH ('TRANSMISSION BIT RATE', 'KBITS/S', BIT RATE, 0, IVAL);
      'IF' IVAL >= 0 'THEN' BIT RATE := IVAL;
      INTPCH ('MAX NUMBER OF DATA BUS TERMINALS', '', TERMINAL LIMIT, 0, IVAL);
      'IF' IVAL >= 0 'THEN' TERMINAL LIMIT := IVAL;
      FLOPCH ('CONTROLLER/TERMINAL OVERHEAD', CONT OVERHEAD, FVAL);
      'IF' FVAL >= 0 'THEN' CONT OVERHEAD := FVAL;
      FLOPCH ('TERMINAL/TERMINAL OVERHEAD', TERM OVERHEAD, FVAL);
      'IF' FVAL >= 0 'THEN' TERM OVERHEAD := FVAL
   'END'
'END' RESET;


'PROCEDURE' INTTYE;
'BEGIN'

   'COMMENT'                                                                    INTTYE (LEVEL 2)
   ***   SAVANT Command.
   ***   Causes a number of entries related to a specific subsystem,
   ***   or to a number of miscellaneous subsystems, to be added to the
   ***   REFLIST from the terminal, with prompts to assist the user.
   ***   Can also be used to create a REFLIST starting from the EMPTY
   ***   state.  For dedicated entries, the subsystem name is read
   ***   first.  This is, therefore, buffered in STRBUFF.  The input
   ***   procedure is then called twice - once for transmitted entries
   ***   and once for received entries.  For miscellaneous entries
   ***   the input procedure is only called once.
   ***   Called by: OBEY COMMAND
   ***   Calls:     CMDWRN          INDAT
   ***              OPTION;

   'IF' STATE <> EMPTY
   'AND' STATE <> OPENX
   'THEN' CMDWRN (1)
   'ELSE'
   'BEGIN'
      'INTEGER' ERROR, SSCODE;

      'COMMENT'
      ***   ERROR: Flag to indicate error in reading.
      ***   SSCODE: Subsystem name code for dedicated entries - set
      ***           in READEN;

      SSCODE := 0; (*** to indicate name not recorded)
      'IF' OPTION ('IF ALL DATA APPERTAINS TO ONE SUBSYSTEM') = TRUE 'THEN'
      'BEGIN'
         'COMMENT' *** dedicated entries;
         ENDLINE(TTO,1);
         WRITE TEXT (TTO, ' SUBSYSTEM NAME .... ');
         INPUT TEXT (TTI);
         INDAT (TTI, 1, SSCODE, ERROR); (*** read tx entries)
         'IF' ERROR = FALSE 'THEN' INDAT (TTI, 0, SSCODE, ERROR);
         (*** read rx entries)
      'END'
      'ELSE' INDAT (TTI, -1, SSCODE, ERROR); (*** miscellaneous data)
      'IF' ERROR = FALSE 'THEN'
      'BEGIN'
```

```
         ENDLINE(TTO,1);
         WRITE TEXT (TTO, ' DATA ENTERED ');
         ENDLINE(TTO,1);
         'IF' ENTRIES > 0 'THEN' STATE := OPENX
      'END'
   'END'
'END' INTTYE;


'PROCEDURE' INFILE;
'BEGIN'

   'COMMENT'                                                                    INFILE (LEVEL 2)
   ***   SAVANT Command.
   ***   Invokes the reading of a disc file of REFLIST entries into the
   ***   EMPTY or OPEN database.  If the state is not originally EMPTY
   ***   then the entries are added to those already there.
   ***   Called by: OBEY COMMAND
   ***   Calls:     CMDWRN            INDAT
   ***              FINFIL;

   'IF' STATE <> EMPTY
   'AND' STATE <> OPENX
   'THEN' CMDWRN (1)
   'ELSE'
   'BEGIN'
      'INTEGER' ERROR, DUMMY;

      'COMMENT'
      ***   ERROR: Flag to indicate error in input.
      ***   DUMMY: Dummy ss code (not needed for misc. entries);

      FINFIL ('DATA', 0, ERROR);
      'IF' ERROR = FALSE 'THEN'
      'BEGIN'
         DUMMY := 0;
         INPUT TEXT (INFIL);
         'IF' TEXT IS ('REFLIST FILE') 'THEN'
         'BEGIN'
            INDAT (INFIL, -1, DUMMY, ERROR);
            CLOSE READ FILE;
            'IF' ERROR = FALSE 'THEN'
            'BEGIN'
               ENDLINE(TTO,1);
               WRITE TEXT (TTO, ' REFLIST READ ');
               ENDLINE(TTO,1);
               STATE := OPENX
            'END'
         'END'
         'ELSE'
         'BEGIN'
            ENDLINE(TTO,1);
            WRITE TEXT (TTO, ' NOT A REFLIST FILE ');
            ENDLINE(TTO,1);
            CLOSE READ FILE
         'END'
      'END'
   'END'
'END' INFILE;


'PROCEDURE' INFILM;
```

```
'BEGIN'

   'COMMENT'                                                                    INFILM (LEVEL 2)
   ***   SAVANT Command.
   ***   Inputs previously filed message data and changes database state
   ***   to LIMITED.
   ***   Called by: OBEY COMMAND
   ***   Calls:     CMDWRN            INNAM
   * +              FINFIL;

   'IF' STATE <> EMPTY
   'THEN' CMDWRN (1)
   'ELSE'
   'BEGIN'
      'INTEGER' CSSPTR, MPTR, MDPTR, LENGTH, ERROR;

      'COMMENT'
      ***   CSSPTR: Pointer to configured subsystem entries.
      ***   MPTR: Pointer to message entries.
      ***   MDPTR: Pointer to data words in message data array.
      ***   LENGTH: Buffer for message length.
      ***   ERROR: Error flag for FINFIL;

      FINFIL ('MESSAGE', 0, ERROR);
      'IF' ERROR = FALSE 'THEN'
      'BEGIN'
         INPUT TEXT (INFIL);
         'IF' TEXT IS ('MESSAGE FILE') 'THEN'
         'BEGIN'
            'COMMENT' *** first input namelists;
            INNAM (SS NAME);
            INNAM (DATA NAME);
            'COMMENT' *** now configured subsystem arrays;
            CONF SS COUNT := READ NUM (INFIL);
            'FOR' CSSPTR := 1 'STEP' 1 'UNTIL' CONF SS COUNT 'DO'
            'BEGIN'
               CONF SS LIST[CSSPTR] := READ NUM (INFIL);
               TERM ADDRESS[CSSPTR] := READ NUM (INFIL);
               SUBADD COUNT[CSSPTR] := 0
            'END';
            'COMMENT' *** now message structure;
            TOTAL MESSAGES := READ NUM (INFIL);
            'FOR' MPTR := 0 'STEP' 1 'UNTIL' TOTAL MESSAGES - 1 'DO'
            'BEGIN'
               TX[MPTR] := READ NUM (INFIL);
               RX[MPTR] := READ NUM (INFIL);
               LENGTH := READ NUM (INFIL);
               RG[MPTR] := LENGTH;
               RETRY[MPTR] := READ NUM (INFIL);
               LENGTH := READ NUM (INFIL);
               WDS[MPTR] := LENGTH;
               'FOR' MDPTR := 1 'STEP' 1 'UNTIL' LENGTH 'DO'
               MESSAGE DATA[MPTR, MDPTR] := READ NUM (INFIL)
            'END';
            CLOSE READ FILE;
            STATE := LIMITED;
            ENDLINE(TTO,1);
            WRITE TEXT (TTO, ' MESSAGES READ ');
            ENDLINE(TTO,1);
         'END'
         'ELSE'
         'BEGIN'
```

```
            ENDLINE(TTO,1);
            WRITE TEXT (TTO, ' NOT A MESSAGE FILE ');
            ENDLINE(TTO,1);
            CLOSE READ FILE
         'END'
      'END'
   'END'
'END' INFILM;


'PROCEDURE' LISENT;
'BEGIN'

   'COMMENT'                                                              LISENT (LEVEL 2)
   ***   SAVANT Command.
   ***   Causes the complete REFLIST to be listed.
   ***   Called by: OBEY COMMAND
   ***   Calls:     CMDWRN            LISREF;

   'IF' STATE <> OPENX
   'AND' STATE <> CONFIGURED
   'AND' STATE <> FORMED
   'AND' STATE <> INCONSISTENT
   'THEN' CMDWRN (1)
   'ELSE' LISREF (0)
'END' LISENT;


'PROCEDURE' LISSSD;
'BEGIN'

   'COMMENT'                                                              LISSSD (LEVEL 2)
   ***   SAVANT Ccmmand.
   ***   Causes all REFLIST entries relating to a particular subsystem
   ***   to be listed.
   ***   Called by: OBEY COMMAND
   ***   Calls:     CMDWRN            LISREF
   ***              LOCNAM;

   'IF' STATE <> OPENX
   'AND' STATE <> CONFIGURED
   'AND' STATE <> FORMED
   'AND' STATE <> INCONSISTENT
   'THEN' CMDWRN (1)
   'ELSE'
   'BEGIN'
      'INTEGER' SSCODE;

      'COMMENT'
      ***   SSCODE: The subsystem name code;

      LOCNAM ("SUBSYSTEM", SS NAME, SSCODE);
      'IF' SSCODE > 0 'THEN' LISREF (SSCODE)
   'END'
'END' LISSSD;


'PROCEDURE' LISSSN;
'BEGIN'

   'COMMENT'                                                              LISSSN (LEVEL 2)
   ***   SAVANT Command.
```

```
      ***   Causes all subsystem names in the SS NAME list to be listed.
      ***   Since all possible names can be contained on one vdu page,
      ***   the option to pause is not offered and is delivered as FALSE
      ***   to the listing procedure.
      ***   Called by: OBEY COMMAND
      ***   Calls:     CMDWRN            LISNAM;

      'IF' STATE = EMPTY
      'THEN' CMDWRN (1)
      'ELSE' LISNAM (SS NAME, FALSE)
   'END' LISSSN;


   'PROCEDURE' LISDNM;
   'BEGIN'

      'COMMENT'                                                          LISDNM (LEVEL 2)
      ***   SAVANT Command.
      ***   Causes all data names in the DATA NAME list to be listed.  Since
      ***   the possible number of names could cause the list to exceed one
      ***   page, the user is offered the pause option which is passed on
      ***   to the listing procedure.
      ***   Called by: OBEY COMMAND
      ***   Calls:     CMDWRN            LISNAM
      ***              OPTION;

      'IF' STATE = EMPTY
      'THEN' CMDWRN (1)
      'ELSE'
      'BEGIN'
         'INTEGER' PAUSE;

         'COMMENT'
         ***   PAUSE: Flag to indicate response to pause option;

         PAUSE := OPTION (EACH PAGE);
         LISNAM (DATA NAME, PAUSE)
      'END'
   'END' LISDNM;


   'PROCEDURE' TRDATA;
   'BEGIN'

      'COMMENT'                                                          TRDATA (LEVEL 2)
      ***   SAVANT Command.
      ***   Causes all REFLIST references to a specified data item to be listed,
      ***   giving the associated subsystem name, whether the item is
      ***   transmitted or received by the subsystem, and the rate, precision
      ***   and units for each reference.
      ***   Called by: OBEY COMMAND
      ***   Calls:     CMDWRN            LOCNAM
      ***              ACTPRC;

      'IF' STATE<>OPENX
      'AND' STATE<>CONFIGURED
      'AND' STATE<>FORMED
      'AND' STATE<>INCONSISTENT
      'THEN' CMDWRN(1)
      'ELSE'
      'BEGIN'
         'INTEGER' DACODE, REFPTR;
```

```
      'COMMENT'
      ***   DACODE: The specific data name code.
      ***   REFPTR: Pointer to the REFLIST entries;

      LOCNAM('DATA',DATA NAME,DACODE);
      'IF' DACODE>0 'THEN'
      'BEGIN'
      ENDLINE(TTO,1);
      OUTPUT TEXT(TTO,'                                    RATE    PREC    UNITS');
      ENDLINE(TTO,1);
      OUTPUT TEXT(TTO,'                                    ----    ----    -----');
         'COMMENT' *** data name located - search throush REFLIST;
         'FOR' REFPTR:=0 'STEP' 1 'UNTIL' ENTRIES-1 'DO'
         'IF' DATA[REFPTR]=DACODE 'THEN'
         'BEGIN'
            'COMMENT' *** relevant entry located - list details;
            ENDLINE(TTO,1);
            NAME(TTO,SS NAME,SS[REFPTR]);
            NAME TAB(TTO,SS NAME,SS[REFPTR],16);
            'IF' TR[REFPTR]=0 'THEN' OUTPUT TEXT(TTO,'RECEIVES ') 'ELSE' OUTPUT TEXT(TTO,'TRANSMITS');
            OUTPUT SPC(TTO,2);
            WRITE NUM(TTO,RATE[REFPTR],2,0);
            OUTPUT SPC(TTO,3);
            WRITE NUM(TTO,ACTPRC(PREC[REFPTR]),4,0);
            OUTPUT SPC(TTO,7);
            NAME(TTO,UNITS NAME,UNITS[REFPTR]);
         'END'
      'END'
   'END'
'END' TRDATA;


'PROCEDURE' CHSSN;
'BEGIN'

   'COMMENT'                                                                    CHSSN (LEVEL 2)
   ***   SAVANT Command.
   ***   Causes a subsystem to have its name changed.
   ***   Called by: OBEY COMMAND
   ***   Calls:     CMDWRN            CHNAM;

   'IF' STATE <> OPENX
   'THEN' CMDWRN (1)
   'ELSE' CHNAM (SS NAME)
'END' CHSSN;


'PROCEDURE' CHDNM;
'BEGIN'

   'COMMENT'                                                                    CHDNM (LEVEL 2)
   ***   SAVANT Command.
   ***   Causes a data item to have its name changed.
   ***   Called by: OBEY COMMAND
   ***   Calls:     CMDWRN            CHNAM;

   'IF' STATE <> OPENX
   'THEN' CMDWRN (1)
   'ELSE' CHNAM (DATA NAME)
'END' CHDNM;
```

```
'PROCEDURE' CHRATE;
'BEGIN'

   'COMMENT'                                                                    CHRATE (LEVEL 2)
   ***   SAVANT Command.
   ***   Causes rate value of particular REFLIST entry to be changed.
   ***   Called by: OBEY COMMAND
   ***   Calls:     CMDWRN            CHNUME;

   'IF' STATE <> OPENX
   'THEN' CMDWRN (1)
   'ELSE' CHNUME (0)
'END' CHRATE;


'PROCEDURE' CHRATG;
'BEGIN'

   'COMMENT'                                                                   CHRATG (LEVEL 2)
   ***   SAVANT Command.
   ***   Causes all rates of a particular value to be changed
   ***   to another value throughout the REFLIST.
   ***   Called by: OBEY COMMAND
   ***   Calls:     CMDWRN            GENNCH;

   'IF' STATE <> OPENX
   'THEN' CMDWRN (1)
   'ELSE' GENNCH (0)
'END' CHRATG;


'PROCEDURE' CHDART;
'BEGIN'

   'COMMENT'                                                                    CHDART (LEVEL 2)
   ***   SAVANT Command.
   ***   Causes the rate value associated with a particular data item
   ***   to be set up at each occurrence of the item on the REFLIST.
   ***   Called by: OBEY COMMAND
   ***   Calls:     CMDWRN            CHDAV;

   'IF' STATE <> OPENX
   'THEN' CMDWRN (1)
   'ELSE' CHDAV (0)
'END' CHDART;


'PROCEDURE' CHSSRT;
'BEGIN'

   'COMMENT'                                                                   CHSSRT (LEVEL 2)
   ***   SAVANT Command.
   ***   Causes all rates of a particular value to be changed to
   ***   another value, but only for the REFLIST entries relating to
   ***   a particular subsystem.
   ***   Called by: OBEY COMMAND
   ***   Calls:     CMDWRN            CHSSV;

   'IF' STATE <> OPENX
   'THEN' CMDWRN (1)
   'ELSE' CHSSV (0)
```

```
'END' CHSSRT;


'PROCEDURE' CHPRCE;
'BEGIN'

   'COMMENT'                                                          CHPRCE (LEVEL 2)
   ***   SAVANT Command.
   ***   Causes precision value of particular REFLIST entry to be changed.
   ***   Called by: OBEY COMMAND
   ***   Calls:     CMDWRN            CHNUME;

   'IF' STATE <> OPENX
   'THEN' CMDWRN (1)
   'ELSE' CHNUME (1)
'END' CHPRCE;


'PROCEDURE' CHPRCG;
'BEGIN'

   'COMMENT'                                                          CHPRCG (LEVEL 2)
   ***   SAVANT Command.
   ***   Causes all precisions of a particular value to be changed
   ***   to another value throughout the REFLIST.
   ***   Called by: OBEY COMMAND
   ***   Calls:     CMDWRN            GENNCH;

   'IF' STATE <> OPENX
   'THEN' CMDWRN (1)
   'ELSE' GENNCH (1)
'END' CHPRCG;


'PROCEDURE' CHDAPR;
'BEGIN'

   'COMMENT'                                                          CHDAPR (LEVEL 2)
   ***   SAVANT Command.
   ***   Causes the precision value associated with a particular data item
   ***   to be set up at each occurrence of the item on the REFLIST.
   ***   Called by: OBEY COMMAND
   ***   Calls:     CMDWRN            CHDAV;

   'IF' STATE <> OPENX
   'THEN' CMDWRN (1)
   'ELSE' CHDAV (1)
'END' CHDAPR;

'PROCEDURE' CHSSPR;
'BEGIN'

   'COMMENT'                                                          CHSSPR (LEVEL 2)
   ***   SAVANT Command.
   ***   Causes all precisions of a particular value to be changed to
   ***   another value; but only for the REFLIST entries relating to
   ***   a particular subsystem.
   ***   Called by: OBEY COMMAND
   ***   Calls:     CMDWRN            CHSSV;

   'IF' STATE <> OPENX
   'THEN' CMDWRN (1)
```

```
   'ELSE' CHSSV (1)
'END' CHSSPR;

'PROCEDURE' CHUNIE;
'BEGIN'

   'COMMENT'                                                                    CHUNIE (LEVEL 2)
   ***   SAVANT Command.
   ***   Changes the units identifier of a specific REFLIST entry, checking
   ***   whether the change has caused the old name to disappear.
   ***   Called by: OBEY COMMAND
   ***   Calls:     CMDWRN        FINENT
   ***              CKREF         RECNAM;

   'IF' STATE<>OPENX
   'THEN' CMDWRN(1)
   'ELSE'
   'BEGIN'
      'INTEGER' LPOS, UNCODE, ERR, DUM1, DUM2;

      'COMMENT'
      ***   LPOS: Pointer to the REFLIST entry.
      ***   UNCODE: The old units name code in the entry.
      ***   ERR: Flag to indicate UNITS NAME list full.
      ***   DUM1: Unused name code.
      ***   DUM2:     ditto;

      FINENT(LPOS,DUM1,DUM2);
      'IF' LPOS>=0 'THEN'
      'BEGIN'
         'COMMENT' *** entry located - save old units name and replace;
        UNCODE:=UNITS[LPOS];
       ENDLINE(TTO,1);
       OUTPUT TEXT(TTO,'NEW UNITS NAME .... ');
        INPUT TEXT(TTI);
        RECNAM(UNITS NAME,LPOS,DUM1,ERR);
        'IF' ERR=FALSE 'THEN'
        'BEGIN'
           'COMMENT' *** check REFLIST for old name;
           CKREF(UNITS NAME,UNCODE);
          ENDLINE(TTO,1);
          OUTPUT TEXT(TTO,'NAME CHANGED');
          ENDLINE(TTO,1);
        'END'
      'END'
   'END'
'END' CHUNIE;


'PROCEDURE' CHUNIG;
'BEGIN'

   'COMMENT'                                                                    CHUNIG (LEVEL 2)
   ***   SAVANT Command.
   ***   Invokes the general change of name of a units identifier.
   ***   Called by: OBEY COMMAND
   ***   Calls:     CMDWRN            CHNAM;

   'IF' STATE <> OPENX
   'THEN' CMDWRN (1)
   'ELSE' CHNAM (UNITS NAME)
'END' CHUNIG;
```

```
'PROCEDURE' CHDAUN;
'BEGIN'

   'COMMENT'
   ***   SAVANT Command.                                          CHDAUN (LEVEL 2)
   ***   Sets up the units identifier associated with a specific data
   ***   item to the specified name at every occurrence of the data
   ***   item in the REFLIST, whatever units identifiers may have
   ***   been associated before.  At each setting, the old units
   ***   name must be checked to see if it has now disappeared from
   ***   the REFLIST.
   ***   Called by: OBEY COMMAND
   ***   Calls:     CMDWRN            CKREF
   ***              RECNAM            LOCNAM;

   'IF' STATE<>OPENX
   'THEN' CMDWRN(1)
   'ELSE'
   'BEGIN'
      'INTEGER' DACODE, REFPTR, UNCODE, ERR, DUMMY;

      'COMMENT'
      ***   DACODE: The specific data name code.
      ***   REFPTR: Pointer to the REFLIST entries.
      ***   UNCODE: The old units name code in each entry.
      ***   ERR: Flag to indicate UNITS NAME list is full.
      ***   DUMMY: Value of new units name code (not needed);

      LOCNAM('DATA',DATA NAME,DACODE);
      'IF' DACODE>0 'THEN'
      'BEGIN'
         'COMMENT' *** data name code located - set required units name;
      ENDLINE(TTO,1);
   OUTPUT TEXT(TTO,'NEW UNITS NAME .... ');
        INPUT TEXT(TTI);
        ERR:=FALSE;
        REFPTR:=0;
        WHILE(REFPTR<ENTRIES 'AND' ERR=FALSE) 'DO'
        'BEGIN'
           'COMMENT' *** work through REFLIST recording the new units name
                     *** code and checking that replaced, at each occurrence
                     *** of the data item;
           'IF' DATA[REFPTR]=DACODE 'THEN'
           'BEGIN'
              'COMMENT' *** entry refers to the data item;
              UNCODE:=UNITS[REFPTR];
              RECNAM(UNITS NAME,REFPTR,DUMMY,ERR);
              CKREF(UNITS NAME,UNCODE)
           'END';
           INC(REFPTR)
        'END';
   'IF' ERR=FALSE 'THEN'
      'BEGIN'ENDLINE(TTO,1);OUTPUT TEXT(TTO,'MOD COMPLETE');
   ENDLINE(TTO,1)
   'END'
   'END'
'END'
'END' CHDAUN;
   'PROCEDURE' DELDAR;
   'BEGIN'

      'COMMENT'                                                   DELDAR (LEVEL 2)
```

```
      ***   SAVANT Command.
      ***   Deletes all REFLIST entries relating to a particular data item,
      ***   closing up the REFLIST to overwrite.  For each entry thus removed,
      ***   the procedure also checks whether this has removed the last
      ***   reference to a subsystem or units name and takes appropriate action.
      ***   Also removes the data name from the DATA NAME namelist and REPOINs
      ***   the REFLIST DATA entries to take account of this.
      ***   Called by: OBEY COMMAND
      ***   Calls:     CMDWRN         DELENT
      ***              CKREF          LOCNAM
      ***              DELNAM         REPOIN;

      'IF' STATE<>OPENX
      'THEN' CMDWRN(1)
      'ELSE'
      'BEGIN'
         'INTEGER' REFPTR, DACODE, SSCODE;

         'COMMENT'
         ***   REFPTR: Pointer to the REFLIST entries.
         ***   SSCODE: The subsystem name code in an entry.
         ***   DACODE: The data name code to be deleted.
         ***   UNCODE: The units name code in an entry;

         LOCNAM('DATA',DATA NAME,DACODE);
         'IF' DACODE>0 'THEN'
         'BEGIN'
            'COMMENT' *** data name code located - find all REFLIST references;
            REFPTR:=0;
            WHILE (REFPTR<ENTRIES) 'DO'
            'IF' DATA[REFPTR]=DACODE 'THEN'
            'BEGIN'
               'COMMENT' *** entry refers to data - delete it and check other names;
               SSCODE:=SS[REFPTR];
               DELENT(REFPTR);
               CKREF(SS NAME,SSCODE)
            'END'
            'ELSE' INC(REFPTR);
           DELNAM (DATA NAME, DACODE); (*** remove from namelist)
           REPOIN(1,DACODE);
          ENDLINE(TTO,1);
          OUTPUT TEXT(TTO,'DATA DELETED');
          ENDLINE(TTO,1);
        'END'
     'END'
'END' DELDAR;


'PROCEDURE' DEL1EN;
'BEGIN'

   'COMMENT'                                                                    DEL1EN (LEVEL 2)
   ***   SAVANT Command.
   ***   Removes one specific entry from the REFLIST, and checks whether
   ***   this action has caused the disappearence from the database
   ***   of any of the names in the entry.
   ***   Called by: OBEY COMMAND
   ***   Calls:     CMDWRN            DELENT
   ***              FINENT            CKREF;

   'IF' STATE<>OPENX
   'THEN' CMDWRN(1)
```

```
   'ELSE'
   'BEGIN'
      'INTEGER' SSCODE, DACODE, UNCODE, LPOS;

      'COMMENT'
      ***   SSCODE: The subsystem name code in the entry.
      ***   DACODE: The data name code in the entry.
      ***   UNCODE: The units name code in the entry.
      ***   LPOS: Pointer to the entry;

      FINENT(LPOS,SSCODE,DACODE);
      'IF' LPOS>=0 'THEN'
      'BEGIN'
         'COMMENT' *** entry exists - delete it;
        UNCODE:=UNITS[LPOS];
        DELENT(LPOS);
        CKREF(SS NAME,SSCODE);
        CKREF(DATA NAME,DACODE);
        CKREF(UNITS NAME,UNCODE);
       ENDLINE(TTO,1);
       OUTPUT TEXT(TTO,"ENTRY DELETED");
       ENDLINE(TTO,1);
      'END'
   'END'
'END' DEL1EN;


'PROCEDURE' FILENT;
'BEGIN'

   'COMMENT'                                                          FILENT (LEVEL 2)
   ***   SAVANT Command.
   ***   Causes the complete REFLIST to be sent to a disc file.
   ***   Called by: OBEY COMMAND
   ***   Calls:     CMDWRN            FILREF;

   'IF' STATE<>OPENX
   'AND' STATE<>CONFIGURED
   'AND' STATE<>FORMED
   'AND' STATE<>INCONSISTENT
   'THEN' CMDWRN(1)
   'ELSE'
   'BEGIN'
      FILREF(0);
      ENDLINE(TTO,1);
      OUTPUT TEXT(TTO,"REFLIST FILED");
      ENDLINE(TTO,1);
   'END'
'END' FILENT;


'PROCEDURE' FILESD;
'BEGIN'

   'COMMENT'                                                        FILESD (LEVEL 2)
   ***   SAVANT Command.
   ***   Causes all REFLIST entries relating to an individual
   ***   subsystem to be sent to a disc file.
   ***   Called by: OBEY COMMAND
   ***   Calls:     CMDWRN            FILREF
   ***              LOCNAM;
```

```
    'IF' STATE<>OPENX
    'AND' STATE<>CONFIGURED
    'AND' STATE<>FORMED
    'AND' STATE<>INCONSISTENT
    'THEN' CMDWRN(1)
    'ELSE'
    'BEGIN'
       'INTEGER' SSCODE;
       'COMMENT'
       ***   SSCODE: The subsystem name code;

       LOCNAM('SUBSYSTEM',SS NAME,SSCODE);
       'IF' SSCODE>0 'THEN'
       'BEGIN'
          FILREF(SSCODE);
          ENDLINE(TTO,1);
          OUTPUT TEXT(TTO,'DATA FILED');
          ENDLINE(TTO,1);
       'END'
    'END'
'END' FILESD;


'PROCEDURE' CLRDB;
'BEGIN'

   'COMMENT'                                                                    CLRDB (LEVEL 2)
   ***   SAVANT Command.
   ***   Returns database state from open or LIMITED to EMPTY,
   ***   initialising the database structures.  The user is requested
   ***   to confirm the intention to clear.
   ***   Called by: OBEY COMMAND
   ***   Calls:     CMDWRN            INIT
   ***              OPTION;

   'IF' STATE <> OPENX
   'AND' STATE <> LIMITED
   'THEN' CMDWRN (1)
   'ELSE''IF' OPTION ('IF YOU REALLY WISH TO CLEAR') = TRUE 'THEN' INIT
   'ELSE' DO NOTHING
'END' CLRDB;
'PROCEDURE' CONFIG;
'BEGIN'

   'COMMENT'                                                                    CONFIG (LEVEL 2)
   ***   SAVANT Command.
   ***   Configures a system from the basic REFLIST by identifying
   ***   the required subsystems, then the appropriate data items and
   ***   their valid receivers, then forming the message structure.
   ***   If fatal inconsistencies exist the database state is set
   ***   to the inconsistent state instead of the configured state.
   ***   Called by: OBEY COMMAND
   ***   Calls:     CMDWRN            FRMTXS
   ***              FRMTXD            FRMRXS
   ***              FATLCK             FRMMES
   ***              FINBSS             PARTITION NUMBER;

   'IF' STATE<>OPENX
   'THEN' CMDWRN(1)
   'ELSE'
   'BEGIN'
      'INTEGER' FATAL;
```

```
      'COMMENT'
      ***   FATAL: Flag to indicate fatal inconsistencies;

      FATAL:=FALSE;
      FRMTXS;
      'IF' CONF SS COUNT>0 'THEN'
      'BEGIN'
         FRMTXD;
         FRMRXS;
         ENDLINE(TTO,1);
         OUTPUT TEXT(TTO,'SYSTEM CONFIGURED');
         ENDLINE(TTO,1);
         FATLCK(FATAL);
         'IF' FATAL=FALSE 'THEN'
         'BEGIN'
            FRMMES;
            FINBSS(FATAL);
            'IF' FATAL=FALSE 'THEN' PARNUM
         'END';
         'IF' FATAL=TRUE 'THEN' 'BEGIN' ENDLINE(TTO,2); OUTPUT TEXT(TTO,'   NO '); 'END'
                                'ELSE' ENDLINE(TTO,1);
         OUTPUT TEXT(TTO,'MESSAGES FORMED');
         ENDLINE(TTO,1);
         STATE:='IF' FATAL=TRUE 'THEN' INCONSISTENT 'ELSE' CONFIGURED
      'END'
   'END'
'END' CONFIG;


'PROCEDURE' LISDAP;
'BEGIN'

   'COMMENT'                                                                    LISDAP (LEVEL 2)
   ***   SAVANT Command.
   ***   For the configured subsystem, this procedure lists all transmitted
   ***   data items.  Each item on the list comprises the data name,
   ***   together with its transmitting subsystem and the associated
   ***   precision and rate capabilities, followed by a list of receiving
   ***   subsystems, together with their precision and rate requirements.
   ***   The option is given to pause on each page, but the pause actually
   ***   occurs before listing the next item which would cause the page
   ***   length to be exceeded.
   ***   Called by: OBEY COMMAND
   ***   Calls:     CMDWRN            OPTION
   ***              ACTPRC;

   'IF' STATE<>CONFIGURED
   'AND' STATE<>FORMED
   'AND' STATE<>INCONSISTENT
   'THEN' CMDWRN(1)
   'ELSE'
   'BEGIN'
      'INTEGER' TXPTR, DATARX, RXPTR, PAUSE, LINES, STOP;
      'COMMENT'
      ***   TXPTR: Pointer to TXLIST entries.
      ***   DATARX: Index to current receiver of data item being examined.
      ***   RXPTR: Pointer to RXLIST entries.
      ***   PAUSE: Flag to indicate response to pause option.
      ***   LINES: Line count.
      ***   STOP: Flag to indicate response to continue option;
```

```
      LINES:=1;
      STOP:=FALSE;
      PAUSE:=OPTION(EACH PAGE);
      'FOR' TXPTR:=0 'STEP' 1 'UNTIL' TOTAL TX DATA-1 'DO'
      'BEGIN'
         'COMMENT' *** work through tx data items;
         'IF' PAUSE=TRUE 'AND' LINES+NRX['TXPTR']+1>24 'THEN'
         'BEGIN'
            'COMMENT' *** check continue option if pausing and page full;
            LINES:=1;
            ENDLINE(TTO,1);
            'IF' OPTION(CONTINUE)=FALSE 'THEN'
            'BEGIN'
               'COMMENT' *** discontinuing;
               TXPTR:=TOTAL TX DATA;
               STOP:=TRUE
            'END'
         'END';
         'IF' STOP=FALSE 'THEN'
         'BEGIN'
            'COMMENT' *** output transmitting info;
            ENDLINE(TTO,1);
            NAME(TTO,DATA NAME,TXDATA['TXPTR']);
            NAME TAB(TTO,DATA NAME,TXDATA['TXPTR'],15);
            OUTPUT TEXT(TTO,"FROM  ");
            NAME(TTO,SS NAME,TXSS['TXPTR']);
            NAME TAB(TTO,SS NAME,TXSS['TXPTR'],24);
            WRITE NUM(TTO,ACTPRC(TXPREC['TXPTR']),4,0);
            OUTPUT TEXT(TTO," BITS AT RATE ");
            WRITE NUM(TTO,TXRATE['TXPTR'],2,0);
            'IF' NRX['TXPTR']=0 'THEN''BEGIN' ENDLINE(TTO,1);  OUTPUT TEXT(TTO,"     NO RECEIVER")
            'END'
            'ELSE'
            'FOR' DATARX:=0 'STEP' 1 'UNTIL' NRX['TXPTR']-1 'DO'
            'BEGIN'
               'COMMENT' *** output info on receivers;
               RXPTR:=RXSTART['TXPTR']+DATARX;
               ENDLINE(TTO,1);
               OUTPUT SPC(TTO,15);
               OUTPUT TEXT(TTO,"TO    ");
               NAME(TTO,SS NAME,RXSS['RXPTR']);
               NAME TAB(TTO,SS NAME,RXSS['RXPTR'],15);
               OUTPUT TEXT(TTO,"REQUIRES ");
               WRITE NUM(TTO,ACTPRC(RXPREC['RXPTR']),4,0);
               OUTPUT TEXT(TTO," BITS AT RATE ");
               WRITE NUM(TTO,RXRATE['RXPTR'],2,0)
            'END';
            LINES:=LINES+NRX['TXPTR']+1
         'END'
      'END'
   'END'
'END' LISDAP;


'PROCEDURE' LISCOS;
'BEGIN'

   'COMMENT'                                                                    LISCOS (LEVEL 2)
   ***   SAVANT Command.
   ***   Produces a list of all configured subsystems, tabulated in
   ***   three columns.  Each name is preceded by a prefix, as follows.
   ***   If the subsystem is not on the bus then the prefix is 'nob'.
```

```
   ***   If the subsystem is on the bus but its terminal address has not
   ***   been set then the prefix is 'uns'; otherwise the prefix is the
   ***   terminal address value.
   ***   Called by: OBEY COMMAND
   ***   Calls:     CMDWRN;

   'IF' STATE<>CONFIGURED
   'AND' STATE<>LIMITED
   'AND' STATE<>FORMED
   'AND' STATE<>LIMITED FORMED
   'AND' STATE<>INCONSISTENT
   'THEN' CMDWRN(1)
   'ELSE'
   'BEGIN'
      'INTEGER' CSSPTR, COLUMN;

      'COMMENT'
      ***   CSSPTR: Pointer to confisured subsystem entries.
      ***   COLUMN: Column count;

      COLUMN:=3;
      'FOR' CSSPTR:=1 'STEP' 1 'UNTIL' CONF SS COUNT 'DO'
      'BEGIN'
         'COMMENT' *** work throush confisured subsystems;
         INC(COLUMN);
         'IF' COLUMN>3 'THEN'
         'BEGIN'
            'COMMENT' *** start in first column;
            COLUMN:=1;
            ENDLINE(TTO,1)
         'END';
         'COMMENT' *** send prefix followed by name, and tab next column;
         OUTPUT TEXT(TTO,"(");
         'IF' TERM ADDRESS[CSSPTR]<0 'THEN' OUTPUT TEXT(TTO,"nob");
         'ELSE''IF' TERM ADDRESS[CSSPTR]=0 'THEN' OUTPUT TEXT(TTO,"uns");
         'ELSE' WRITE NUM(TTO,TERM ADDRESS[CSSPTR],2,0);
         OUTPUT TEXT(TTO,") ");
          NAME(TTO,SS NAME,CONF SS LIST[CSSPTR]);
         'IF' COLUMN<3 'AND' CSSPTR<CONF SS COUNT
         'THEN' NAME TAB(TTO,SS NAME,CONF SS LIST[CSSPTR],17);
      'END';
      'COMMENT' *** send count of confisured subsystems;
      ENDLINE(TTO,2);
      WRITE NUM(TTO,CONF SS COUNT,3,0);
      OUTPUT TEXT(TTO," CONFIGURED SUBSYSTEMS");
      ENDLINE(TTO,1)
   'END'
'END' LISCOS;


'PROCEDURE' LISDAT;
'BEGIN'

   'COMMENT'                                                          LISDAT (LEVEL 2)
   ***   SAVANT Command.
   ***   This procedure produces a list of all source/sink pairs in
   ***   the confisured system, tosether with details of the data
   ***   passins between each.  The user is offered the option to
   ***   pause after each s/s pair listed.
   ***   Called by: OBEY COMMAND
   ***   Calls:     CMDWRN            LISSSP
   ***              OPTION;
```

```
    'IF' STATE <> CONFIGURED
    'AND' STATE <> FORMED
    'THEN' CMDWRN (1)
    'ELSE'
    'BEGIN'
       'INTEGER' TXCODE, RXCODE, PAUSE, DONE;

       'COMMENT'
       ***   TXCODE: Pointer to source on CONF SS LIST.
       ***   RXCODE: Pointer to sink on CONF SS LIST.
       ***   PAUSE: Flag to indicate response to pause option.
       ***   DONE: Flag to indicate traffic found for s/s pair;

       PAUSE := OPTION ('TO PAUSE AFTER EACH SOURCE/SINK PAIR');
       'FOR' TXCODE := 1 'STEP' 1 'UNTIL' CONF SS COUNT 'DO'
       'FOR' RXCODE := 1 'STEP' 1 'UNTIL' CONF SS COUNT 'DO'
       'IF' TXCODE <> RXCODE 'THEN'
       'BEGIN'
          'COMMENT' *** work through source/sink pairs;
          LISSSP (TXCODE, RXCODE, DONE);
          'IF' PAUSE = TRUE 'AND' DONE = TRUE 'THEN'
          'BEGIN'
             'COMMENT' *** check continue OPTION if pausing and output has occurred;
             'IF' OPTION (CONTINUE) = FALSE 'THEN'
             'BEGIN'
                'COMMENT' *** discontinuing - set pointers to escape;
                TXCODE := CONF SS COUNT;
                RXCODE := CONF SS COUNT
             'END'
          'END'
       'END'
    'END'
'END' LISDAT;


'PROCEDURE' MAPDAT;
'BEGIN'

   'COMMENT'
   ***   SAVANT Command.                                                    MAPDAT (LEVEL 2)
   ***   Produces a visual map of data traffic between source/sink pairs.
   ***   The map is formed as a matrix, using the MAP array, which is
   ***   first primed by SETMAP.  Taking each row as representing a
   ***   configured subsystem acting as source, and each column as
   ***   representing a configured subsystem acting as sink, the message
   ***   list is scanned and an appropriate letter character placed
   ***   at the appropriate element of the matrix to indicate the
   ***   type of traffic: B - bus traffic only, D - direct traffic
   ***   only, and C - both direct and bus traffic.  When it comes
   ***   to deciding on a C, the direct transfers will have been determined
   ***   first since they are earlier in the message list.  Thus, if
   ***   a bus transfer is discovered for a particular source/sink pair
   ***   then the appropriate element can be examined.  If it already
   ***   contains a D, then a C is inserted, otherwise a B is inserted.
   ***   To output the map, the user is given an option.  It can either
   ***   be displayed on the terminal or it can be sent to a disc file
   ***   for future listing.  The latter alternative is attractive if
   ***   there are many subsystems (eg >20) since the map would then be
   ***   too large to display on a terminal screen.  During output, the
   ***   corresponding subsystem name is listed for each row, so that
   ***   the numbers labelling the columns can be identified.
```

```
***   Called by: OBEY COMMAND
***   Calls:     CMDWRN              SETMAP
***              FINFIL              OPTION
***              CLISTR;

'IF' STATE <> CONFIGURED
'AND' STATE <> FORMED
'AND' STATE <> LIMITED
'AND' STATE <> LIMITED FORMED
'THEN' CMDWRN (1)
'ELSE'
'BEGIN'
   'INTEGER' TXCODE, RXCODE, MPTR, DEST, ERROR;

   'COMMENT'
   ***   TXCODE: Pointer to source on CONF SS LIST.
   ***   RXCODE: Pointer to sink on CONF SS LIST.
   ***   MPTR: Pointer to message entries.
   ***   DEST: Destination device.
   ***   ERROR: Dummy error flag for FINFIL;

   SETMAP;
   'FOR' MPTR := 0 'STEP' 1 'UNTIL' TOTAL MESSAGES - 1 'DO'
   'BEGIN'
      'COMMENT' *** work through message list taking each source/sink
                *** pair as encountered;
      TXCODE := CLISTR (TX[MPTR]);
      RXCODE := CLISTR (RX[MPTR]);
      'IF' RG[MPTR] <> 0 'THEN'
         'BEGIN'
            'IF' MAP[TXCODE, RXCODE] <> 195 'THEN'
             MAP[TXCODE, RXCODE] := ('IF' MAP[TXCODE, RXCODE] = 196 'THEN' 195 'ELSE' 194)
         'END'
      'ELSE' MAP[TXCODE, RXCODE] := 196
   'END';
   'IF' OPTION ("TO SEND MAP TO FILE") = TRUE 'THEN'
   'BEGIN'
      'COMMENT' *** destination is disc file;
      FINFIL ("MAP", 1, ERROR);
      WRITE TEXT (OUTFIL, "MAP FILE");
      ENDLINE(TTO,1);
      DEST := OUTFIL
   'END'
   'ELSE' DEST := TTO; (*** destination is terminal)
   'FOR' TXCODE := -1 'STEP' 1 'UNTIL' CONF SS COUNT 'DO'
   'BEGIN'
      'COMMENT' *** send each row of map array to destination;
      NEWLINE (DEST, 1);
      'IF' TXCODE > 0 'THEN'
      'BEGIN'
         'COMMENT' *** send subsystem name;
         NAME (DEST, SS NAME, CONF SS LIST[TXCODE]);
         NAME TAB (DEST, SS NAME, CONF SS LIST[TXCODE], 17)
      'END'
      'ELSE' SPACES (DEST, 17);
      'FOR' RXCODE := -1 'STEP' 1 'UNTIL' CONF SS COUNT 'DO'
      'BEGIN'
         'COMMENT' *** send row elements;
         'IF' RXCODE > 0 'THEN' SEND CHAR (DEST, 'OCTAL'(240));
         SEND CHAR (DEST, MAP[TXCODE, RXCODE])
      'END'
   'END';
```

```
            'IF' DEST <> TTO 'THEN'
            'BEGIN'
               ENDLINE(TTO,2);
               WRITE TEXT (TTO, 'MAPPING COMPLETE');
               ENDLINE(TTO,1);
               NEWLINE (OUTFIL, 1);
               CLOSE WRITE FILE
            'END'
         'END'
      'END' MAPDAT;


      'PROCEDURE' SSPRDA;
      'BEGIN'

         'COMMENT'                                                                    SSPRDA (LEVEL 2)
         ***   SAVANT Command.
         ***   Produces a list of all data passing between one source/sink pair.
         ***   Called by: OBEY COMMAND
         ***   Calls:     CMDWRN            LISSSP
         ***              LOCNAM            CLISTR;

         'IF' STATE <> CONFIGURED
         'AND' STATE <> FORMED
         'THEN' CMDWRN (1)
         'ELSE'
         'BEGIN'
            'INTEGER' TXCODE, RXCODE, DONE, ERRMESS;

            'COMMENT'
            ***   TXCODE: Pointer to source on CONF SS LIST.
            ***   RXCODE: Pointer to sink on CONF SS LIST.
            ***   DONE: Flag to indicate traffic exists.
            ***   ERRMESS: Pointer to error message string;

            ERRMESS := '
SUBSYSTEM NOT CONFIGURED
';
            LOCNAM ('SOURCE SUBSYSTEM', SS NAME, TXCODE);
            'IF' TXCODE > 0 'THEN'
            'BEGIN'
               'COMMENT' *** source exists - find it on CONF SS LIST;
               TXCODE := CLISTR (TXCODE);
               'IF' TXCODE > CONF SS COUNT 'THEN' WRITE TEXT (TTO, ERRMESS)
               'ELSE'
               'BEGIN'
                  'COMMENT' *** found it - locate sink;
                  LOCNAM ('SINK SUBSYSTEM', SS NAME, RXCODE);
                  'IF' RXCODE > 0 'THEN'
                  'BEGIN'
                     'COMMENT' *** sink exists - find it on CONF SS LIST;
                     RXCODE := CLISTR (RXCODE);
                     'IF' RXCODE > CONF SS COUNT 'THEN' WRITE TEXT (TTO, ERRMESS)
                     'ELSE''IF' TXCODE = RXCODE 'THEN'
                     'BEGIN'
                        ENDLINE(TTO,1);
                        WRITE TEXT (TTO, 'SAME SINK AS SOURCE');
                        ENDLINE(TTO,1);
                     'END'
                     'ELSE'
                     'BEGIN'
                        'COMMENT' *** found it - list traffic;
```

```
                    LISSSP (TXCODE, RXCODE, DONE);
                    'IF' DONE = FALSE 'THEN'
                    'BEGIN'
                       ENDLINE(TTO,1);
                       WRITE TEXT (TTO, 'NO DATA');
                       ENDLINE(TTO,1)
                    'END'
                 'END'
              'END'
           'END'
        'END'
     'END'
'END' SSPRDA;


'PROCEDURE' CKCONS;
'BEGIN'

   'COMMENT'                                                          CKCONS (LEVEL 2)
   ***   SAVANT Command.
   ***   Determines the fatal inconsistencies in the inconsistent system.
   ***   If there are no fatal data inconsistencies, it concludes that there
   ***   must be too many potential bus terminals.
   ***   Called by: OBEY COMMAND
   ***   Calls:     CMDWRN            FATLCK;

   'IF' STATE <> INCONSISTENT
   'THEN' CMDWRN (1)
   'ELSE'
   'BEGIN'
      'INTEGER' FATAL;

      'COMMENT'
      ***   FATAL: Flag to confirm fatal data inconsistencies;

      FATAL := FALSE;
      FATLCK (FATAL);
      'IF' FATAL = FALSE 'THEN'
      'BEGIN'
         ENDLINE(TTO,1);
         WRITE TEXT (TTO, 'NO FATAL DATA INCONSISTENCIES - ');
         WRITE TEXT (TTO, 'TOO MANY TERMINALS');
         ENDLINE(TTO,1)
      'END'
   'END'
'END' CKCONS;


'PROCEDURE' CKCORR;
'BEGIN'

   'COMMENT'                                                          CKCORR (LEVEL 2)
   ***   SAVANT Command.
   ***   Performs a check that all input and output requirements in the
   ***   configured system are satisfied, reporting the details of any
   ***   deficiencies.  Firstly examines the NRX entries of TXLIST in order
   ***   to determine whether any transmitted data items have no configured
   ***   receivers.  Then scans the CONF field of the REFLIST to check if
   ***   any received entries involving configured subsystems have not
   ***   been dealt with, which implies a received data item requirement
   ***   which is not supplied by any configured transmitter.
   ***   Called by: OBEY COMMAND
```

```
   ***   Calls:     CMDWRN;

   'IF' STATE <> CONFIGURED
   'AND' STATE <> FORMED
   'AND' STATE <> INCONSISTENT
   'THEN' CMDWRN (1)
   'ELSE'
   'BEGIN'
      'INTEGER' TXPTR, REFPTR, CSSPTR;

      'COMMENT'
      ***   TXPTR: Pointer to TXLIST entries.
      ***   REFPTR: Pointer to the REFLIST entries.
      ***   CSSPTR: Pointer to confisured subsystem entries;

      NEWLINE (TTO, 1);
      'FOR' TXPTR := 0 'STEP' 1 'UNTIL' TOTAL TX DATA - 1 'DO'
      'IF' NRX[TXPTR] = 0 'THEN'
      'BEGIN'
         'COMMENT' *** no receiver for tx data;
         NAME (TTO, DATA NAME, TXDATA[TXPTR]);
         WRITE TEXT (TTO, ' FROM ');
         NAME (TTO, SS NAME, TXSS[TXPTR]);
         WRITE TEXT (TTO, ' NOT USED');
       ENDLINE(TTO,1);
      'END';
      'FOR' REFPTR := 0 'STEP' 1 'UNTIL' ENTRIES - 1 'DO'
      'IF' CONF[REFPTR] = FALSE 'THEN'
      'BEGIN'
         'COMMENT' *** entry not dealt with - see if subsystem confisured;
         CSSPTR := 1;
         WHILE (CSSPTR <= CONF SS COUNT) 'DO'
         'LEGIN'
            'IF' SS[REFPTR] = CONF SS LIST[CSSPTR] 'THEN'
            'BEGIN'
               'COMMENT' *** subsystem confisured - data not supplied;
               NAME (TTO, DATA NAME, DATA[REFPTR]);
               WRITE TEXT (TTO, ' TO ');
               NAME (TTO, SS NAME, SS[REFPTR]);
               WRITE TEXT (TTO, ' NOT SUPPLIED');
               ENDLINE(TTO,1);
               CSSPTR := CONF SS COUNT
            'END';
            INC (CSSPTR)
         'END'
      'END';
      ENDLINE(TTO,1);
      WRITE TEXT (TTO,' CHECK COMPLETE');
      ENDLINE(TTO,1);
   'END'
'END' CKCORR;


'PROCEDURE' LISMES;
'BEGIN'

   'COMMENT'                                                          LISMES (LEVEL 2)
   ***   SAVANT Command.
   ***   Causes details of all bus messages to be listed.
   ***   Called by: OBEY COMMAND
   ***   Calls:     CMDWRN          LISMDA;
```

```
   'IF' STATE <> CONFIGURED
   'AND' STATE <> LIMITED
   'AND' STATE <> FORMED
   'AND' STATE <> LIMITED FORMED
   'THEN' CMDWRN (1)
   'ELSE' LISMDA (1)
'END' LISMES;


'PROCEDURE' LISDDA;
'BEGIN'

   'COMMENT'                                                                    LISDDA (LEVEL 2)
   ***   SAVANT Command.
   ***   Causes details of all direct data transfers to be listed.
   ***   Called by: OBEY COMMAND
   ***   Calls:     CMDWRN            LISMDA;

   'IF' STATE <> CONFIGURED
   'AND' STATE <> LIMITED
   'AND' STATE <> FORMED
   'AND' STATE <> LIMITED FORMED
   'THEN' CMDWRN (1)
   'ELSE' LISMDA (0)
'END' LISDDA;


'PROCEDURE' CALLOA;
'BEGIN'

   'COMMENT'                                                                    CALLOA (LEVEL 2)
   ***   SAVANT Command.
   ***   This procedure calculates the percentage bus loadings which
   ***   would be achieved were the configured system implemented in
   ***   practice, using the system parameters for word and message
   ***   overheads, word length and transmission rate.  The user specifies
   ***   which subsystem is bus controller - if a name which is not one
   ***   of the configured subsystems is given then a dedicated controller
   ***   is assumed - and also specifies the maximum iteration rate,
   ***   which is that represented by Rate 1.  The procedure first
   ***   assembles the number of words, including message overheads,
   ***   associated with each rate group, and also updates several running
   ***   counts in the process.  AVLOAD assembles the sum of messages
   ***   times rate for use in average bus loading calculation, MAXLOAD
   ***   assembles a sum of all messages, for use in peak lumped loading
   ***   calculations, and MAXDIS is the number of messages at whichever
   ***   rate other than Rate 1 involves the most messages, for use in the
   ***   peak distributed loading calculations.  After listing the
   ***   individual rate load figures, the procedure then calculates and
   ***   displays these bus loading percentages.
   ***   Called by: OBEY COMMAND
   ***   Calls:     CMDWRN            FINBC;

   'IF' STATE <> CONFIGURED
   'AND' STATE <> LIMITED
   'AND' STATE <> FORMED
   'AND' STATE <> LIMITED FORMED
   'THEN' CMDWRN (1)
   'ELSE'
   'BEGIN'
      'INTEGER' MPTR, NDIV, RGRP, BCU;
      'FLOATING' MAXRATE, MAXLOAD, AVLOAD, MAXDIS, LOAD, FACTOR,
```

```
ACTUAL RATE, OHEAD, LENGTH;
'FLOATING''ARRAY' RATELOAD[1:15];

'COMMENT'
***   MPTR: Pointer to message entries.
***   NDIV: Division counter for determining actual rate.
***   RGRP: The current rate group.
***   BCU: The name code of the bus controller (0 if dedicated).
***   MAXRATE: The maximum iteration rate in Hz.
***   MAXLOAD: See main comment.
***   AVLOAD:      ditto
***   MAXDIS:      ditto
***   LOAD: The current individual rate load, for listing & comparison.
***   FACTOR: Multiplication factor for forming loading percentages.
***   ACTUAL RATE: The current actual rate in Hz.
***   OHEAD: The overhead for the current message.
***   LENGTH: The number of words in the message (inc overheads).
***   RATELOAD: Array for individual rate load counts;

FINBC (BCU);
ENDLINE(TTO,1);
WRITE TEXT (TTO, "MAX ITERATION RATE .... ");
MAXRATE := READ NUM (TTI);
MAXLOAD := 0;
AVLOAD := 0;
'COMMENT' *** initialise RATELOAD array;
'FOR' RGRP := 1 'STEP' 1 'UNTIL' LOWEST RATE 'DO' RATELOAD[RGRP] := 0;
'FOR' MPTR := 0 'STEP' 1 'UNTIL' TOTAL MESSAGES - 1 'DO'
'BEGIN'
   'COMMENT' *** work through message list, updating appropriate sums;
   RGRP := RG[MPTR];
   'IF' RGRP > 0 'THEN'
   'BEGIN'
      'COMMENT' *** message is on the bus;
      ACTUAL RATE := 2*MAXRATE;
      'FOR' NDIV := 1 'STEP' 1 'UNTIL' RGRP 'DO' ACTUAL RATE := ACTUAL RATE/2;
      OHEAD := 'IF' TX[MPTR] = BCU 'OR' RX[MPTR] = BCU
            'THEN' CONT OVERHEAD 'ELSE' TERM OVERHEAD;
      LENGTH := OHEAD + WDS[MPTR];
      AVLOAD := AVLOAD + LENGTH*ACTUAL RATE;
      MAXLOAD := MAXLOAD + LENGTH;
      RATELOAD[RGRP] := RATELOAD[RGRP] + LENGTH
   'END'
'END';
'COMMENT' *** list loads at each rate and form MAXDIS;
ENDLINE(TTO,2);
WRITE TEXT (TTO, "LOAD AT EACH RATE:");
ENDLINE(TTO,2);
ACTUAL RATE := 2*MAXRATE;
MAXDIS := 0;
'FOR' RGRP := 1 'STEP' 1 'UNTIL' LOWEST RATE 'DO'
'BEGIN'
   ACTUAL RATE := ACTUAL RATE/2;
   LOAD := RATELOAD[RGRP];
   WRITE NUM (TTO, ACTUAL RATE, 4, 3);
   WRITE TEXT (TTO, " HZ:   ");
   WRITE NUM (TTO, LOAD, 4, 3);
   WRITE TEXT (TTO, " WORDS.");
   ENDLINE(TTO,1);
   'IF' RGRP > 1 'AND' LOAD > MAXDIS 'THEN' MAXDIS := LOAD
'END';
'COMMENT' *** form multiplication factor to convert word sums into
```

```
            *** percentage time loadings and calculate percentages;
      FACTOR := 'FLOATING'(WORD LENGTH + WORD OVERHEAD)/'FLOATING'(10*BIT RATE);
      ENDLINE(TTO,1);
      WRITE TEXT (TTO, 'AVERAGE BUS LOADING  =  ');
      WRITE NUM (TTO, FACTOR*AVLOAD, 3, 2);
      WRITE TEXT (TTO, ' PERCENT');
      ENDLINE(TTO,2);
      WRITE TEXT (TTO, 'PEAK MINOR CYCLE LOADS:');
      ENDLINE(TTO,1);
      WRITE TEXT (TTO, '        DISTRIBUTED  =  ');
      WRITE NUM (TTO, FACTOR*MAXRATE*(RATELOAD[1] + MAXDIS), 3, 2);
      WRITE TEXT (TTO, ' PERCENT');
      ENDLINE(TTO,1);
      WRITE TEXT (TTO, '        LUMPED       =  ');
      WRITE NUM (TTO, FACTOR*MAXRATE*MAXLOAD, 3, 2);
      WRITE TEXT (TTO, ' PERCENT');
   'END'
'END' CALLOA;


'PROCEDURE' SUMMES;
'BEGIN'

   'COMMENT'                                                                    SUMMES (LEVEL 2)
   ***   SAVANT Command.
   ***   Causes a summary list of all messages in the configured system
   ***   to be displayed.  The listing pauses after each page, with the
   ***   option to discontinue.  For each message the procedure tabulates
   ***   the user message number, source and sink subsystems, rate group,
   ***   retry code and number of words.  For direct transfers, the rate
   ***   is displayed as 'D' and no retry code is output.
   ***   Called by: OBEY COMMAND
   ***   Calls:     CMDWRN          OPTION;

   'IF' STATE <> CONFIGURED
   'AND' STATE <> LIMITED
   'AND' STATE <> FORMED
   'AND' STATE <> LIMITED FORMED
   'THEN' CMDWRN (1)
   'ELSE'
   'BEGIN'
      'INTEGER' MPTR, LINES;

      'COMMENT'
      ***   MPTR: Pointer to message entries.
      ***   LINES: Line count;

      LINES := 1;
      'FOR' MPTR := 0 'STEP' 1 'UNTIL' TOTAL MESSAGES - 1 'DO'
      'BEGIN'
         'COMMENT' *** work through message list;
         'IF' LINES = 1 'THEN'
         'BEGIN'
            'COMMENT' *** title columns at start of page;
            ENDLINE(TTO,1);
            WRITE TEXT (TTO, '    No.   FROM             TO                 RATE  RETRY  WORDS');
            ENDLINE(TTO,1);
            WRITE TEXT (TTO, '    ---   ----             --                 ----  -----  -----');
            ENDLINE(TTO,1);
         'END';
         'COMMENT' *** list info;
         WRITE NUM (TTO, MPTR+1, 3, 0);
```

```
            WRITE TEXT (TTO, '    ');
            NAME (TTO, SS NAME, TX[MPTR]);
            NAME TAB (TTO, SS NAME, TX[MPTR], 17);
            NAME (TTO, SS NAME, RX[MPTR]);
            NAME TAB (TTO, SS NAME, RX[MPTR], 17);
            'IF' RG[MPTR] = 0
            'THEN' WRITE TEXT (TTO, '     D       ')
            'ELSE'
            'BEGIN'
               WRITE NUM (TTO, RG[MPTR], 5, 0);
               WRITE TEXT(TTO,' ');
               WRITE NUM (TTO, RETRY[MPTR], 6, 0)
            'END';
            WRITE NUM (TTO, WDS[MPTR], 6, 0);
            NEWLINE (TTO, 1);
            INC (LINES);
            'IF' LINES = 21 'THEN'
            'BEGIN'
               'COMMENT' *** check continue option if page full;
               LINES := 1;
               'IF' OPTION (CONTINUE) = FALSE 'THEN' MPTR := TOTAL MESSAGES
            'END'
         'END'
      'END'
'END' SUMMES;


'PROCEDURE' LIS1ME;
'BEGIN'

   'COMMENT'                                                                    LIS1ME (LEVEL 2)
   ***   SAVANT Command.
   ***   Causes the contents of a single bus message to be listed.
   ***   Called by: OBEY COMMAND
   ***   Calls:     CMDWRN            FINMES
   ***              SENMES;

   'IF' STATE <> CONFIGURED
   'AND' STATE <> LIMITED
   'AND' STATE <> FORMED
   'AND' STATE <> LIMITED FORMED
   'THEN' CMDWRN (1)
   'ELSE'
   'BEGIN'
      'INTEGER' MPTR;

      'COMMENT'
      ***   MPTR: Pointer to message to be listed;

      FINMES ('MESSAGE NUMBER', MPTR);
      'IF' MPTR >= 0 'THEN' SENMES (MPTR)
   'END'
'END' LIS1ME;


'PROCEDURE' SETRET;
'BEGIN'

   'COMMENT'                                                                    SETRET (LEVEL 2)
   ***   SAVANT Command.
   ***   Sets retry codes for bus messages.  The procedure has two
   ***   modes of operation.  In the first mode the user can set the
```

```
***   same code in all messages except those called out specifically
***   as exceptions.  In requesting the exception message numbers
***   the user is repeatedly prompted until the number of a message
***   which does not exist is given (typically 0).  The EXCEPT field
***   of the MESSAGE HEADER table is used to mark exception messages.
***   In the second mode, the user sets the code value in one
***   specific message.
***   Called by: OBEY COMMAND
***   Calls:     CMDWRN            FINMES
***              FINCOD               OPTION;

'IF' STATE <> CONFIGURED
'AND' STATE <> LIMITED
'THEN' CMDWRN (1)
'ELSE'
'BEGIN'
   'INTEGER' MPTR, CODE, OK;

   'COMMENT'
   ***   MPTR: Pointer to message entries.
   ***   CODE: The retry code value.
   ***   OK: Flag to indicate exceptions still being read;

   'IF' OPTION ('FOR GENERAL RETRY CODE SET') = TRUE 'THEN'
   'BEGIN'
      'COMMENT' *** general setting - initialise except elements;
      'FOR' MPTR := 0 'STEP' 1 'UNTIL' TOTAL MESSAGES - 1 'DO' EXCEPT[MPTR] := FALSE;
      OK := TRUE;
      WHILE (OK = TRUE) 'DO'
      'BEGIN'
         'COMMENT' *** set exception number and mark message;
         FINMES ('EXCEPTION MESSAGE NUMBER', MPTR);
         'IF' MPTR >= 0
         'THEN' EXCEPT[MPTR] := TRUE
         'ELSE'
         'BEGIN'
            'COMMENT' *** non-existent message - end of exceptions;
            WRITE TEXT (TTO, 'END OF EXCEPTIONS');
            ENDLINE(TTO,1);
            OK := FALSE
         'END'
      'END';
      'COMMENT' *** set code and record in all appropriate messages;
      FINCOD (CODE);
      'FOR' MPTR := 0 'STEP' 1 'UNTIL' TOTAL MESSAGES - 1 'DO'
      'IF' EXCEPT[MPTR] = FALSE 'THEN' RETRY[MPTR] := CODE;
      ENDLINE(TTO,1);
      WRITE TEXT (TTO, 'CODES SET');
      ENDLINE(TTO,1);
   'END'
   'ELSE'
   'BEGIN'
      'COMMENT' *** individual setting;
      FINMES ('MESSAGE NUMBER', MPTR);
      'IF' MPTR >= 0 'THEN'
      'BEGIN'
         'COMMENT' *** message exists - set code and record;
         FINCOD (CODE);
         RETRY[MPTR] := CODE;
         ENDLINE(TTO,1);
         WRITE TEXT (TTO, 'CODE SET');
         ENDLINE(TTO,1);
```

```
         'END'
      'END'
   'END'
'END' SETRET;


'PROCEDURE' SETTAD;
'BEGIN'
   'COMMENT'                                                                    SETTAD (LEVEL 2)
   ***   SAVANT Command.
   ***   Sets terminal addresses, either for all configured subsystems
   ***   or for that specified.  If all are being set, any valid ta value
   ***   can be assigned except one already used in the current setting.
   ***   If an individual ta is being set, no value already assigned
   ***   can be used.
   ***   Called by: OBEY COMMAND
   ***   Calls:     CMDWRN            CKTA
   ***              OPTION            LOCNAM
   ***              CONF SS REF;

   'IF' STATE <> CONFIGURED
   'AND' STATE <> LIMITED
   'THEN' CMDWRN (1)
   'ELSE'
   'BEGIN'
      'INTEGER' REPLY, SSCODE, CSSPTR, ERROR;

      'COMMENT'
      ***   REPLY: TA value from user.
      ***   SSCODE: Subsystem name code for individual setting.
      ***   CSSPTR: Pointer to configured subsystem entries.
      ***   ERROR: Flag to indicate erroneous reply;

      'IF' OPTION ('IF ALL TA'S ARE TO BE SET') = TRUE 'THEN'
      'BEGIN'
         'COMMENT' *** general setting;
         ENDLINE(TTO,1);
         WRITE TEXT (TTO, 'TYPE TA VALUE AFTER SUBSYSTEM NAME');
         ENDLINE(TTO,1);
         CSSPTR := 1;
         WHILE (CSSPTR <= CONF SS COUNT) 'DO'
         'BEGIN'
            'COMMENT' *** work through configured subsystems;
            'IF' TERM ADDRESS[CSSPTR] < 0 'THEN' INC (CSSPTR)
            'ELSE'
            'BEGIN'
               'COMMENT' *** subsystem is on bus - offer name and set
                         *** valid reply;
               ENDLINE(TTO,1);
               NAME (TTO, SS NAME, CONF SS LIST[CSSPTR]);
               WRITE TEXT (TTO, ' .... ');
               REPLY := READ NUM (TTI);
               CKTA (CSSPTR, REPLY, CSSPTR, ERROR);
               'IF' ERROR = FALSE 'THEN'
               'BEGIN'
                  'COMMENT' *** record reply;
                  TERM ADDRESS[CSSPTR] := REPLY;
                  INC (CSSPTR)
               'END'
            'END'
         'END'
```

```
      'END'
      'ELSE'
      'BEGIN'
         'COMMENT' *** individual setting;
         LOCNAM ('SUBSYSTEM', SS NAME, SSCODE);
         'IF' SSCODE > 0 'THEN'
         'BEGIN'
            'COMMENT' *** subsystem exists - find it on CONF SS LIST;
            CSSPTR := CLISTR (SSCODE);
            'IF' CSSPTR > CONF SS COUNT
            'THEN'
            'BEGIN'
               ENDLINE(TTO,1);
               WRITE TEXT (TTO, 'THIS SUBSYSTEM NOT CONFIGURED');
               ENDLINE(TTO,1);
            'END'
            'ELSE'
            'BEGIN'
               'COMMENT' *** see if it's on the bus;
               'IF' TERM ADDRESS[CSSPTR] < 0
               'THEN'
               'BEGIN'
                  ENDLINE(TTO,1);
                  WRITE TEXT (TTO, 'THIS SUBSYSTEM IS NOT ON THE BUS');
                  ENDLINE(TTO,1);
               'END'
               'ELSE'
               'BEGIN'
                  'COMMENT' *** set valid ta value and record;
                  ERROR := TRUE;
                  WHILE (ERROR = TRUE) 'DO'
                  'BEGIN'
                     WRITE TEXT (TTO, 'TA VALUE .... ');
                     REPLY := READ NUM (TTI);
                     CKTA (CSSPTR, REPLY, CONF SS COUNT, ERROR)
                  'END';
                  TERM ADDRESS[CSSPTR] := REPLY
               'END'
            'END'
         'END'
      'END'
   'END'
'END' SETTAD;


'PROCEDURE' LISSUB;
'BEGIN'                                                          LISSUB (LEVEL 2)

   'COMMENT'
   ***   SAVANT Command.
   ***   Causes message subsets to be formed and marked, listing
   ***   the results.  The option is given to pause on each page.
   ***   Called by: OBEY COMMAND
   ***   Calls:     CMDWRN              FRMSUB
   ***              OPTION;

   'IF' STATE <> CONFIGURED
   'AND' STATE <> LIMITED
   'AND' STATE <> FORMED
   'AND' STATE <> LIMITED FORMED
   'THEN' CMDWRN (1)
   'ELSE'
```

```
      'BEGIN'
         'INTEGER' MPTR, PAUSE, LINES;

         'COMMENT'
         ***   MPTR: Pointer to message entries.
         ***   PAUSE: Flag to indicate response to pause option.
         ***   LINES: Line count;

         PAUSE := OPTION (EACH PAGE);
         FRMSUB;
         LINES := 1;
         'FOR' MPTR := 0 'STEP' 1 'UNTIL' TOTAL MESSAGES - 1 'DO'
         'BEGIN'
            'COMMENT' *** work through message list;
            'IF' SUBSET[MPTR] > 0 'THEN'
            'BEGIN'
               'COMMENT' *** message is a subset - list info;
               WRITE TEXT (TTO, 'MESSAGE');
               WRITE NUM (TTO, MPTR+1, 3, 0);
               WRITE TEXT (TTO, ' IS A SUBSET OF MESSAGE');
               WRITE NUM (TTO, SUBSET[MPTR], 3, 0);
               NEWLINE (TTO, 1);
               INC (LINES)
            'END';
            'IF' PAUSE = TRUE 'AND' LINES = 23 'THEN'
            'BEGIN'
               'COMMENT' *** check continue option if pausing and page full;
               LINES := 1;
               'IF' OPTION (CONTINUE) = FALSE 'THEN' MPTR := TOTAL MESSAGES
            'END'
         'END'
      'END'
   'END' LISSUB;


   'PROCEDURE' REORDM;
   'BEGIN'

      'COMMENT'
      ***   SAVANT Command.                                                    REORDM (LEVEL 2)
      ***   This procedure effects the reordering of data words in a message.
      ***   It moves a specified block of data within a message to a new
      ***   specified position.
      ***   Called by: OBEY COMMAND
      ***   Calls:     CMDWRN            FINMES
      ***              SENMES            MODABL;

      'IF' STATE <> CONFIGURED
      'AND' STATE <> LIMITED
      'THEN' CMDWRN (1)
      'ELSE'
      'BEGIN'
         'INTEGER' MPTR, LOW, HIGH, NEW, BLOCK, SIZE;

         'COMMENT'
         ***   MPTR: Pointer to message being reordered.
         ***   LOW: Pointer to first word of block being moved.
         ***   HIGH: Pointer to last word of block being moved.
         ***   NEW: Pointer to new position of first word.
         ***   BLOCK: Number of words in data block being moved.
         ***   SIZE: Length of message being reordered;
```

```
      FINMES ('MESSAGE TO BE REORDERED', MPTR);
      'IF' MPTR >= 0 'THEN'
      'BEGIN'
         'COMMENT' *** located message to be reordered - display its contents
                   *** and locate data block;
         SENMES (MPTR);
         SIZE := WDS[MPTR];
         ENDLINE(TTO,1);
         WRITE TEXT (TTO, 'LOW POINTER  .... ');
         LOW := READ NUM (TTI);
         ENDLINE(TTO,1);
         WRITE TEXT (TTO, 'HIGH POINTER .... ');
         HIGH := READ NUM (TTI);
         ENDLINE(TTO,1);
         WRITE TEXT (TTO, 'NEW POSITION .... ');
         NEW := READ NUM (TTI);
         'IF' LOW > SIZE 'OR' HIGH > SIZE 'OR' LOW > HIGH
         'THEN'
         'BEGIN'
            ENDLINE(TTO,1);
            WRITE TEXT (TTO, 'POINTERS INCONSISTENT');
            ENDLINE(TTO,1);
         'END'
         'ELSE'
         'BEGIN'
            'COMMENT' *** pointers ok - check new position;
            BLOCK := HIGH - LOW + 1;
            'IF' NEW + BLOCK - 1 > SIZE
            'THEN'
            'BEGIN'
               ENDLINE(TTO,1);
               WRITE TEXT (TTO, 'SORRY - WONT FIT');
               ENDLINE(TTO,1);
            'END'
            'ELSE'
            'BEGIN'
               'COMMENT' *** move data block;
               MODABL (MPTR, MPTR, LOW, NEW, BLOCK);
               ENDLINE(TTO,1);
               WRITE TEXT (TTO, 'REORDERING COMPLETE');
               ENDLINE(TTO,1);
               SENMES (MPTR)
            'END'
         'END'
      'END'
   'END'
'END' REORDM;


'PROCEDURE' REFMTM;
'BEGIN'

   'COMMENT'                                                          REFMTM (LEVEL 2)
   ***   SAVANT Command.
   ***   This procedure performs the reformatting task.  It removes a
   ***   specified block of data from a yielding message and deposits
   ***   it either as a new message or in another existing message,
   ***   known as the accepting message, provided that the source, sink
   ***   and rate of the accepting message are identical to those of the
   ***   yielding message.
   ***   Called by: OBEY COMMAND
   ***   Calls:     CMDWRN            FINMES
```

```
***              OPELIS              CLOLIS
***              OPTION              SENMES
***              MODABL;

'IF' STATE <> CONFIGURED
'AND' STATE <> LIMITED
'THEN' CMDWRN (1)
'ELSE'
'BEGIN'
   'INTEGER' YMPTR, AMPTR, BLOCK, LOW, HIGH, SIZE, LIST OPENED;

   'COMMENT'
   ***   YMPTR: Pointer to message yielding data.
   ***   AMPTR: Pointer to message accepting data.
   ***   BLOCK: Number of words in data block being moved.
   ***   LOW: Pointer to first word of block being moved.
   ***   HIGH: Pointer to last word of block being moved.
   ***   SIZE: Number of words in yielding or accepting message.
   ***   LIST OPENED: Flag to indicate void message created;

   FINMES ("MESSAGE YIELDING DATA", YMPTR);
   'IF' YMPTR >= 0 'THEN'
   'BEGIN'
      'COMMENT' *** yielding message identified;
      'IF' OPTION ("IF YIELDED PORTION IS TO CREATE A NEW MESSAGE") = TRUE 'THEN'
      'BEGIN'
         'COMMENT' *** new message to be created - open a gap at the next
                       *** message entry to accept the data;
         AMPTR := YMPTR + 1;
         OPELIS (YMPTR);
         LIST OPENED := TRUE
      'END'
      'ELSE'
      'BEGIN'
         'COMMENT' *** accepting message already exists - locate it;
         FINMES ("MESSAGE ACCEPTING DATA", AMPTR);
         LIST OPENED := FALSE
      'END';
      'IF' AMPTR >= 0 'THEN'
      'BEGIN'
         'COMMENT' *** check parameters of two messages match;
         'IF' TX[AMPTR] <> TX[YMPTR]
         'OR' RX[AMPTR] <> RX[YMPTR]
         'OR' RG[AMPTR] <> RG[YMPTR]
         'THEN'
         'BEGIN'
           ENDLINE(TTO,1);
           WRITE TEXT (TTO, "MESSAGE PARAMETERS DONT MATCH");
           ENDLINE(TTO,1);
         'END'
         'ELSE'
         'BEGIN'
            'COMMENT' *** display yielding message and identify block
                      *** of data to be moved;
            SIZE := WDS[YMPTR];
            SENMES (YMPTR);
            ENDLINE(TTO,1);
            WRITE TEXT (TTO, "LOW POINTER  .... ");
            LOW := READ NUM (TTI);
            ENDLINE(TTO,1);
            WRITE TEXT (TTO, "HIGH POINTER .... ");
            HIGH := READ NUM (TTI);
```

```
                'IF' LOW > SIZE 'OR' HIGH > SIZE 'OR' LOW > HIGH 'THEN'
                'BEGIN'
                   'COMMENT' *** pointers supplied dont tally;
                   ENDLINE(TTO,1);
                   WRITE TEXT (TTO, 'POINTERS INCONSISTENT');
                   ENDLINE(TTO,1);
                   'IF' LIST OPENED = TRUE 'THEN' CLOLIS (AMPTR)
                'END'
                'ELSE'
                'BEGIN'
                   'COMMENT' *** check accepting message can accommodate data;
                   BLOCK := HIGH - LOW + 1;
                   'IF' WDS[AMPTR] + BLOCK > MESSAGE LENGTH
                   'THEN'
                   'BEGIN'
                      ENDLINE(TTO,1);
                      WRITE TEXT (TTO, 'ACCEPTING MESSAGE CANT ACCOMMODATE');
                      ENDLINE(TTO,1);
                   'END'
                   'ELSE'
                   'BEGIN'
                      'COMMENT' *** move the data;
                      MODABL (YMPTR, AMPTR, LOW, 1, BLOCK);
                      ENDLINE(TTO,1);
                      WRITE TEXT (TTO, 'REFORMATTING COMPLETE');
                      ENDLINE(TTO,1);
                      'COMMENT' *** display reformatted messages;
                      'IF' WDS[YMPTR] > 0
                      'THEN' SENMES (YMPTR)
                      'ELSE'
                      'BEGIN'
                         'COMMENT' *** yielding message now empty;
                         CLOLIS (YMPTR);
                         'IF' AMPTR > YMPTR 'THEN' DEC (AMPTR)
                      'END';
                      SENMES (AMPTR)
                   'END'
                'END'
             'END'
          'END'
       'END'
    'END'
'END' REFMTM;


'PROCEDURE' FILMES;
'BEGIN'

   'COMMENT'                                                              FILMES (LEVEL 2)
   ***   SAVANT Command.
   ***   This procedure sends the configured system message structure and
   ***   supporting arrays to a specified file.
   ***   Called by: OBEY COMMAND
   ***   Calls:     CMDWRN            FILNAM
   ***              FINFIL               FILNUM;

   'IF' STATE <> CONFIGURED
   'AND' STATE <> LIMITED
   'AND' STATE <> FORMED
   'AND' STATE <> LIMITED FORMED
   'THEN' CMDWRN (1)
   'ELSE'
```

```
   'BEGIN'
      'INTEGER' CSSPTR, MPTR, MDPTR, SIZE, ERROR;

      'COMMENT'
      ***   CSSPTR: Pointer to configured subsystem entries.
      ***   MPTR: Pointer to message entries.
      ***   MDPTR: Pointer to data words in message data array.
      ***   SIZE: Buffer for message length.
      ***   ERROR: Dummy error flag for FINFIL;

      FINFIL ('MESSAGE', 1, ERROR);
      WRITE TEXT (OUTFIL, 'MESSAGE FILE');
      ENDLINE(OUTFIL,1);
      'COMMENT' *** first file required namelists;
      FILNAM (SS NAME);
      FILNAM (DATA NAME);
      FILNUM (CONF SS COUNT);
      'FOR' CSSPTR := 1 'STEP' 1 'UNTIL' CONF SS COUNT 'DO'
      'BEGIN'
         'COMMENT' *** file configured subsystem arrays;
         FILNUM (CONF SS LIST[CSSPTR]);
         FILNUM (TERM ADDRESS[CSSPTR])
      'END';
      FILNUM (TOTAL MESSAGES);
      'FOR' MPTR := 0 'STEP' 1 'UNTIL' TOTAL MESSAGES - 1 'DO'
      'BEGIN'
         'COMMENT' *** file message header and contents for each message;
         FILNUM (TX[MPTR]);
         FILNUM (RX[MPTR]);
         FILNUM (RG[MPTR]);
         FILNUM (RETRY[MPTR]);
         SIZE := WDS[MPTR];
         FILNUM (SIZE);
         'FOR' MDPTR := 1 'STEP' 1 'UNTIL' SIZE 'DO' FILNUM (MESSAGE DATA[MPTR, MDPTR])
      'END';
      CLOSE WRITE FILE;
      ENDLINE(TTO,1);
      WRITE TEXT (TTO, 'MESSAGES FILED');
      ENDLINE(TTO,1);
   'END'
'END' FILMES;


'PROCEDURE' DISMAN;
'BEGIN'

   'COMMENT'                                                                    DISMAN (LEVEL 2)
   ***   SAVANT Command.
   ***   This procedure restores the configured database to the open state,
   ***   initialising affected database structures.
   ***   Called by: OBEY COMMAND
   ***   Calls:     CMDWRN            CLRCFL
   ***              CLRMEL;

   'IF' STATE <> CONFIGURED
   'AND' STATE <> INCONSISTENT
   'THEN' CMDWRN (1)
   'ELSE'
   'BEGIN'
      'INTEGER' REFPTR;

      'COMMENT'
```

```
      ***   REFPTR: Pointer to the REFLIST entries;

      CLRMEL;
      CLRCFL;
      'FOR' REFPTR := 1 'STEP' 1 'UNTIL' ENTRIES - 1 'DO' CONF[REFPTR] := FALSE;
      ENDLINE(TTO,1);
      WRITE TEXT (TTO, 'SYSTEM DISMANTLED');
      ENDLINE(TTO,1);
      STATE := OPENX
   'END'
'END' DISMAN;


'PROCEDURE' FRMSYS;
'BEGIN'

   'COMMENT'                                                                    FRMSYS (LEVEL 2)
   ***   SAVANT Command.
   ***   This procedure is used to form the configured system.  It checks
   ***   that all terminal addresses have been set, and if this is so,
   ***   forms subsets, then subaddresses.  Provided subaddress forming
   ***   succeeds, the database state is changed.
   ***   Called by: OBEY COMMAND
   ***   Calls:     CMDWRN            FRMSUB
   ***              CKTA ST           FRMSAD;

   'IF' STATE <> CONFIGURED
   'AND' STATE <> LIMITED
   'THEN' CMDWRN (1)
   'ELSE'
   'BEGIN'
      'INTEGER' ERROR;

      'COMMENT'
      ***   ERROR: Flag to indicate error in forming;

      CKTA ST (ERROR);
      'IF' ERROR = TRUE 'THEN'
      'BEGIN'
         ENDLINE(TTO,1);
         WRITETEXT(TTO,'ALL TA'S NOT SET');
         ENDLINE(TTO,1);
      'END'
      'ELSE'
      'BEGIN'
         FRMSUB;
         FRMSAD (ERROR);
         'IF' ERROR = FALSE 'THEN'
         'BEGIN'
            ENDLINE(TTO,1);
            WRITE TEXT (TTO, 'SYSTEM FORMED');
            ENDLINE(TTO,1);
            STATE := 'IF' STATE = CONFIGURED 'THEN' FORMED 'ELSE' LIMITED FORMED
         'END'
      'END'
   'END'
'END' FRMSYS;


'PROCEDURE' LISSAD;
'BEGIN'
```

```
'COMMENT'                                                                    LISSAD (LEVEL 2)
***   SAVANT Command.
***   This procedure produces a list of transmitted and received
***   subaddress assignments for each bus message in the formed system.
***   The user is offered the option to pause on each page and to
***   discontinue listing after any page.
***   Called by: OBEY COMMAND
***   Calls:     CMDWRN            OPTION;

'IF' STATE <> FORMED
'AND' STATE <> LIMITED FORMED
'THEN' CMDWRN (1)
'ELSE'
'BEGIN'
   'INTEGER' MPTR, PAUSE, LINES;

   'COMMENT'
   ***   MPTR: Pointer to message entries.
   ***   PAUSE: Flag to indicate response to pause option.
   ***   LINES: Line count;

   PAUSE := OPTION (EACH PAGE);
   LINES := 1;
   'COMMENT' *** work through message list;
   'FOR' MPTR := 0 'STEP' 1 'UNTIL' TOTAL MESSAGES - 1 'DO'
   'IF' RG[MPTR] <> 0 'THEN'
   'BEGIN'
      'COMMENT' *** message is on bus - LISSAD;
      WRITE NUM (TTO, MPTR+1, 3, 0);
      OUTPUT SPC(TTO,4);
      WRITE TEXT (TTO, "SA");
      WRITE NUM (TTO, TXSA[MPTR], 2, 0);
      WRITE TEXT (TTO, " IN ");
      NAME (TTO, SS NAME, TX[MPTR]);
      NAME TAB (TTO, SS NAME, TX[MPTR], 17);
      WRITE TEXT (TTO, "TO SA");
      WRITE NUM (TTO, RXSA[MPTR], 2, 0);
      WRITE TEXT (TTO, " IN ");
      NAME (TTO, SS NAME, RX[MPTR]);
      NAME TAB (TTO, SS NAME, RX[MPTR], 17);
      WRITE NUM (TTO, WDS[MPTR], 2, 0);
      WRITE TEXT (TTO, " WORDS");
      ENDLINE(TTO,1);
      INC (LINES);
      'IF' PAUSE = TRUE 'AND' LINES = 23 'THEN'
      'BEGIN'
         'COMMENT' *** check continue option if pausing and page full;
         LINES := 1;
         'IF' OPTION (CONTINUE) = FALSE 'THEN' MPTR := TOTAL MESSAGES
      'END'
   'END'
'END' LISSAD;


'PROCEDURE' FILSCH;
'BEGIN'

   'COMMENT'                                                                 FILSCH (LEVEL 2)
   ***   SAVANT Command.
   ***   For the formed system, this procedure causes a list of bus control
   ***   schedules to be sent to a specified file, and a list of subaddress
```

```
   ***   contents and direct data requirements for each subsystem to another.
   ***   Called by: OBEY COMMAND
   ***   Calls:     CMDWRN       FILTAB
   ***              FILSAD       FINFIL;

   'IF' STATE <> FORMED
   'AND' STATE <> LIMITED FORMED
   'THEN' CMDWRN (1)
   'ELSE'
   'BEGIN'
      'INTEGER' ERROR;

      'COMMENT'
      ***   ERROR: Dummy error flag for FINFIL;

      FINFIL ("SCHEDULE", 1, ERROR);
      WRITE TEXT (OUTFIL, "SCHEDULE FILE");
      ENDLINE(OUTFIL,1);
      FILTAB;
      CLOSE WRITE FILE;
      FINFIL ("LISTING", 1, ERROR);
      WRITE TEXT (OUTFIL, "INFO FILE");
      ENDLINE(OUTFIL,1);
      FILSAD;
      CLOSE WRITE FILE;
      ENDLINE(TTO,1);
      WRITE TEXT (TTO, "SCHEDULES FILED");
      ENDLINE(TTO,1);
   'END'
'END' FILSCH;


'PROCEDURE' UNFSYS;
'BEGIN'

   'COMMENT'                                                    UNFSYS - FORM SYSTEM (LEVEL 2)
   ***   SAVANT command.
   ***   Returns database to appropriate configured state
   ***   from one of the formed states, deleting the
   ***   subaddress assignments.
   ***   Called by: OBEY COMMAND
   ***   Calls:     CMDWRN;

   'IF' STATE<>FORMED
   'AND' STATE<>LIMITED FORMED
   'THEN' CMDWRN(1)
   'ELSE'
   'BEGIN'
      'INTEGER' MPTR, CSSPTR;
      'FOR' MPTR:=0 'STEP' 1 'UNTIL' TOTAL MESSAGES-1 'DO'
      'BEGIN'
         TXSA[MPTR]:=0;
         RXSA[MPTR]:=0
      'END';
      'FOR' CSSPTR:=1 'STEP' 1 'UNTIL' CONF SS COUNT 'DO' SUBADD COUNT[CSSPTR]:=0;
      ENDLINE(TTO,1);
      OUTPUT TEXT(TTO,"SYSTEM UNFORMED");
      ENDLINE(TTO,1);
      STATE:='IF' STATE=FORMED 'THEN' CONFIGURED 'ELSE' LIMITED
   'END'
'END' UNFSYS;
```

```
'PROCEDURE' INIT;
'BEGIN'

   'COMMENT'                                                                    INIT (LEVEL 3)
   ***   Returns the database to the EMPTY state, initialising counts
   ***   and all database tables and arrays.
   ***   Called by: SAVD21          CLRDB
   ***   Calls:     LISCMD          CLRCFL
   ***              CLRMEL;

   'INTEGER' PTR1, PTR2;

   'COMMENT'
   ***   PTR1: Pointer used as index for clearing REFLIST and namelists.
   ***   PTR2: Pointer used as index to individual namelist words;

   CLRMEL;
   CLRCFL;
   'FOR' PTR1 := 0 'STEP' 1 'UNTIL' 3*MAXENTRIES - 1 'DO' REFLIST[PTR1] := 0;
   ENTRIES := 0;
   'FOR' PTR2 := 0 'STEP' 1 'UNTIL' 6 'DO'
   'BEGIN'
      'FOR' PTR1 := 0 'STEP' 1 'UNTIL' MAXSS 'DO'
      'BEGIN'
         SS NAME[PTR1, PTR2] := 0;
         UNITS NAME[PTR1, PTR2] := 0
      'END';
      'FOR' PTR1 := 0 'STEP' 1 'UNTIL' MAXDATA 'DO' DATA NAME[PTR1, PTR2] := 0
   'END';
   SS NAME[0, 1] := MAXSS;
   SS NAME[0, 2] := 0;
   DATA NAME[0, 1] := MAXDATA;
   DATA NAME[0, 2] := 1;
   UNITS NAME[0, 1] := MAXSS;
   UNITS NAME[0, 2] := 2;
   STATE := EMPTY;
   LISCMD
'END' INIT;
'PROCEDURE' CMDWRN ('VALUE''INTEGER' TYPE);
'BEGIN'

   'COMMENT'                                                                    CMDWRN (LEVEL 3)
   ***   Warns of invalid command and reminds of listing option.
   ***   If TYPE is delivered as 0, the command is completely
   ***   unknown, whereas if TYPE is 1 then the command has been
   ***   recognised but the user has attempted to invoke it in
   ***   a state in which it is not valid.  The erroneous command is
   ***   repeated back to the user in the warning message.
   ***   Called by: OBEY COMMAND and all Level 2 procedures;

   ENDLINE(TTO,1);
   OUTPUT TEXT(TTO,"COMMAND ");
   OUTPUT TEXT(TTO,STRING(STRBUFF));
   'IF' TYPE=0 'THEN'
'BEGIN' OUTPUT TEXT(TTO," NOT UNDERSTOOD");
   ENDLINE(TTO,1)
   'END'
   'ELSE'
   'BEGIN'
    OUTPUT TEXT(TTO," NOT VALID IN THIS STATE");
    ENDLINE(TTO,1)
```

```
    'END';
      ENDLINE(TTO,1);
      OUTPUT TEXT(TTO,'TYPE 'H' TO LIST VALID COMMANDS');
      ENDLINE(TTO,1);
'END' CMDWRN;


'PROCEDURE' INTPCH
   ('VALUE''INTEGER' PARAM, UNITS, IDENT, LIMIT; 'LOCATION''INTEGER' VALUE);
'BEGIN'

   'COMMENT'                                                                    INTPCH (LEVEL 3)
   ***   Offers the user the option of changing one of the integer
   ***   preset parameters.  If the option is accepted the new value
   ***   is returned as the parameter VALUE.  If the option to change
   ***   is not accepted, VALUE is returned negative.  This means that
   ***   a new value which is input negative will not be recognised
   ***   by the calling procedure.  PARAM is a text string identifying
   ***   the parameter in question, IDENT is the current value, UNITS is a
   ***   text string giving the units in which the quantity is represented
   ***   and LIMIT is the maximum allowable value.  If there is no limit,
   ***   LIMIT is delivered as 0.
   ***   Called by: RESET
   ***   Calls:     OPTION;

   'INTEGER' REPLY;

   'COMMENT'
   ***   REPLY: Buffer for reply;

   ENDLINE(TTO,1);
   OUTPUT TEXT(TTO,PARAM);
   OUTPUT TEXT(TTO,' VALUE IS');
   WRITE NUM(TTO,IDENT,4,0);
   OUTPUT SPC(TTO,1); OUTPUT TEXT(TTO,UNITS);
   'IF' OPTION('IF YOU WISH TO CHANGE')=TRUE 'THEN'
   'BEGIN'
      'COMMENT' *** change required - GETNV;
     ENDLINE(TTO,1);
      OUTPUT TEXT(TTO,'TYPE NEW VALUE .... ');
      REPLY:=READ NUM(TTI);
      'IF' LIMIT>0 'THEN'
      'BEGIN'
         'COMMENT' *** if there is a limit, check the value;
         WHILE(REPLY>LIMIT) 'DO'
         'BEGIN'
            'COMMENT' *** reply over limit - keep trying;
     ENDLINE(TTO,1);
            OUTPUT TEXT(TTO,'LIMIT IS');
            WRITE NUM(TTO,LIMIT,2,0);
            OUTPUT TEXT(TTO,', NOW THEN .... ');
            REPLY:=READ NUM(TTI)
         'END'
      'END'
   'END'
   'ELSE' REPLY:=-1;
   VALUE:=REPLY
'END' INTPCH;


'PROCEDURE' FLOPCH
   ('VALUE''INTEGER' PARAM; 'VALUE''FLOATING' IDENT; 'LOCATION''FLOATING' VALUE);
```

```
'BEGIN'

   'COMMENT'                                                                    FLOPCH (LEVEL 3)
   ***   Offers the user the option of changing one of the floating
   ***   preset parameters.  If the option is accepted the new value
   ***   is returned as the parameter VALUE.  If the option to change
   ***   is not accepted, VALUE is returned negative.  This means that
   ***   a new value which is input negative will not be recognised
   ***   by the calling procedure.  PARAM is a text string identifying
   ***   the parameter in question and IDENT is the current value.
   ***   Called by: RESET
   ***   Calls:     OPTION;

   ENDLINE(TTO,1);
   OUTPUT TEXT(TTO,PARAM);
   OUTPUT TEXT(TTO,' VALUE IS');
   WRITE NUM(TTO,IDENT,1,1);
   OUTPUT TEXT(TTO,' WORDS');
   'IF' OPTION('IF YOU WISH TO CHANGE')=TRUE 'THEN'
   'BEGIN'
      'COMMENT' *** change required - GETNV;
     ENDLINE(TTO,1);
      OUTPUT TEXT(TTO,'TYPE NEW VALUE .... ');
      VALUE:=READ NUM(TTI)
   'END'
   'ELSE' VALUE:=-1
'END' FLOPCH;


'PROCEDURE' INDAT
   ('VALUE''BYTE' CHNL;'VALUE''INTEGER'  TRBIT; 'LOCATION''INTEGER' SSCODE, ERR);
'BEGIN'

   'COMMENT'                                                                    INDAT (LEVEL 3)
   ***   Inputs a number of entries to the REFLIST, either from a disc
   ***   file or from the terminal, depending on the value of CHNL.
   ***   Ascertains the number required either by prompting the user or
   ***   by reading the first file item.  If the number to be read would
   ***   cause the REFLIST table bounds to be exceeded, a warning is sent
   ***   and no input takes place.  The value of TRBIT is set by the
   ***   calling procedure.  If it is -1 then this indicates that
   ***   miscellaneous entries must be read.  If it is not negative
   ***   then the procedure is expected to read a number of entries
   ***   related to a single subsystem, whose name has been read and
   ***   is buffered in STRBUFF.  A value of 1 indicates that transmitted
   ***   entries are to be read and 0, received entries.  Each entry
   ***   is actually input by the procedure READEN.  SSCODE is the
   ***   name code of the subsystem for the single subsystem type of input.
   ***   Its value is zeroed in the calling procedure, and it is passed on
   ***   to the procedure READEN for setting.  ERR is used to
   ***   indicate that the input process has been aborted.  This
   ***   procedure uses it to indicate too many entries, and it is
   ***   passed on to READEN for that procedure to indicate
   ***   a full namelist.
   ***   Called by: INTTYE  INFILE
   ***   Calls:     READEN;

   'INTEGER' NUMBER, RDCOUNT;

   'COMMENT'
   ***   NUMBER: Number of entries to be read.
   ***   RDCOUNT: Running count of entries read;
```

```
   'COMMENT' *** Find how many entries to be read;
   'IF' CHNL=TTI 'THEN'
   'BEGIN'
     ENDLINE(TTO,1);
       OUTPUT TEXT(TTO,'HOW MANY ');
     'IF' TRBIT<0 'THEN' OUTPUT TEXT(TTO,'ENTRIES .... ')
     'ELSE''IF' TRBIT=1 'THEN' OUTPUT TEXT(TTO,'TRANSMITTED ENTRIES .... ')
     'ELSE' OUTPUT TEXT(TTO,'RECEIVED ENTRIES .... ')
   'END';
   NUMBER:=READ NUM(CHNL);
   'IF' ENTRIES+NUMBER>MAXENTRIES 'THEN'
   'BEGIN'
      'COMMENT' *** this is too many - warn and dont read;
     ERR:=TRUE;
     ENDLINE(TTO,1);
     OUTPUT TEXT(TTO,'***ERROR***  ONLY ROOM FOR ');
     WRITE NUM(TTO,MAXENTRIES-ENTRIES,4,0);
     OUTPUT TEXT(TTO,' ENTRIES');
     ENDLINE(TTO,2);
   'END'
   'ELSE'
   'BEGIN'
      'COMMENT' *** otherwise, read the entries;
     RDCOUNT:=1;
     ERR:=FALSE;
     WHILE (RDCOUNT<=NUMBER) 'DO'
     'BEGIN'
        READEN(CHNL,TRBIT,SSCODE,ERR);
        'IF' ERR=FALSE 'THEN' INC(ENTRIES) 'ELSE' RDCOUNT:=NUMBER;
        INC(RDCOUNT)
     'END'
   'END'
'END' INDAT;


'PROCEDURE' INNAM ('BYTE''ARRAY' NAMELIST);
'BEGIN'

   'COMMENT'                                                          INNAM (LEVEL 3)
   ***   During the input of filed messages, inputs a namelist from
   ***   a file to which it has been sent in whole word form by the
   ***   procedure FILNAM.
   ***   Called by: INFILM;

   'INTEGER' NLPTR, STWDPTR, SIZE;

   'COMMENT'
   ***   NLPTR: Pointer to namelist entries.
   ***   STWDPTR: Pointer to individual strins words.
   ***   SIZE: Buffer for namelist size;

   SIZE := READ NUM (INFIL);
   NAMELIST[0, 0] := SIZE;
   'FOR' NLPTR := 1 'STEP' 1 'UNTIL' SIZE 'DO'
   'FOR' STWDPTR := 0 'STEP' 1 'UNTIL' 6 'DO'
   NAMELIST[NLPTR, STWDPTR] := READ NUM (INFIL)
'END' INNAM;
'PROCEDURE' LISREF ('VALUE''INTEGER' CODE);
'BEGIN'

   'COMMENT'                                                          LISREF (LEVEL 3)
```

```
***   Lists either partial or complete REFLIST.  If CODE is 0 the
***   complete REFLIST is listed, grouped subsystem by subsystem,
***   otherwise only those entries relating to the subsystem whose
***   code is the value of CODE are listed.  The option is given to
***   pause after each page of the listing, and to discontinue
***   listing after any page.  If the listing runs to completion
***   then a count of entries is displayed.
***   Called by: LISENT            LISSSD
***   Calls:     OPTION            ACTPRC;

'INTEGER' REFPTR, ENTRYCOUNT, SSCODE, LOWER, UPPER, LINES, PAUSE, STOP;

'COMMENT'
***   REFPTR: Pointer to the REFLISENT.
***   ENTRYCOUNT: Count of entries listed.
***   SSCODE: The current subsystem name code.
***   LOWER: Lower bound subsystem code for output loop.
***   UPPER: Upper bound subsystem code for output loop.
***   LINES: Line count.
***   PAUSE: Flag to indicate response to pause option.
***   STOP: Flag to indicate response to continue option;

PAUSE:=OPTION(EACH PAGE);
LINES:=1;
STOP:=FALSE;
ENTRYCOUNT:=0;
'COMMENT' *** set lower and upper subsystem bounds for listing;
LOWER:='IF' CODE=0 'THEN' 1 'ELSE' CODE;
UPPER:='IF' CODE=0 'THEN' SS NAME[0,0] 'ELSE' CODE;
'FOR' SSCODE:=LOWER 'STEP' 1 'UNTIL' UPPER 'DO'
'FOR' REFPTR:=0 'STEP' 1 'UNTIL' ENTRIES-1 'DO'
'IF' SS[REFPTR]=SSCODE 'THEN'
'BEGIN'
   'COMMENT' *** for each subsystem in turn, list relevant entries;
   'IF' LINES=1 'THEN'
   'BEGIN'
      'COMMENT' *** title columns at beginning of listing and at
                *** beginning of each page if pausing;
  ENDLINE(TTO,1);
  OUTPUT TEXT(TTO,'SUBSYSTEM        DATA            T/R          RATE        PREC     UNITS');
  ENDLINE(TTO,1);
  OUTPUT TEXT(TTO,'---------        ----            ---          ----        ----     -----');
  ENDLINE(TTO,1);
   'END';
   NAME(TTO,SS NAME,SSCODE);
   NAME TAB(TTO,SS NAME,SSCODE,17);
   NAME(TTO,DATA NAME,DATA[REFPTR]);
   NAME TAB(TTO,DATA NAME,DATA[REFPTR],17);
   'IF' TR[REFPTR]=1 'THEN' OUTPUT TEXT(TTO,'T') 'ELSE' OUTPUT TEXT(TTO,'R');
   OUTPUT SPC(TTO,9);
   WRITE NUM(TTO,RATE[REFPTR],2,0);
   OUTPUT SPC(TTO,6);
   WRITE NUM(TTO,ACTPRC(PREC[REFPTR]),4,0);
   OUTPUT SPC(TTO,5);
   NAME(TTO,UNITS NAME,UNITS[REFPTR]);
   ENDLINE(TTO,1);
   INC(LINES);
   'IF' PAUSE=TRUE 'AND' LINES=21 'THEN'
   'BEGIN'
      'COMMENT' *** check continue option if pausing and page full;
      LINES:=1;
      'IF' OPTION(CONTINUE)=FALSE 'THEN'
```

```
            'BEGIN'
               'COMMENT' *** discontinuing listing;
               REFPTR:=ENTRIES;
               SSCODE:=UPPER;
               STOP:=TRUE
            'END'
         'END';
         INC(ENTRYCOUNT)
      'END';
      ENDLINE(TTO,1);
      'IF' STOP=FALSE 'THEN'
      'BEGIN'
         'COMMENT' *** if appropriate send count;
         WRITE NUM(TTO,ENTRYCOUNT,4,0);
         OUTPUT TEXT(TTO,' ENTRIES');
        ENDLINE(TTO,1);
      'END'
   'END' LISREF;


   'PROCEDURE' LISNAM ('BYTE''ARRAY' NAMELIST; 'VALUE''INTEGER' PAUSE);
   'BEGIN'

      'COMMENT'                                                                    LISNAM (LEVEL 3)
      ***   Lists the names on the appropriate namelist in five columns.
      ***   PAUSE specifies whether the option to pause on each page has
      ***   been accepted in the calling procedure.  If the list is allowed
      ***   to run to completion then the count of names is also sent.
      ***   Called by: LISSSN          LISDNM
      ***   Calls:     OPTION;

      'INTEGER' NLPTR, LIST TYPE, SIZE, COLUMN, LINES, STOP;

      'COMMENT'
      ***   NLPTR: Pointer to namelist entries.
      ***   LIST TYPE: Buffer for namelist type.
      ***   SIZE: Buffer for namelist size.
      ***   COLUMN: Column count.
      ***   LINES: Line count.
      ***   STOP: Flag to indicate response to continue option;

      LINES:=1;
      STOP:=FALSE;
      SIZE:=NAMELIST[0,0];
      LIST TYPE:=NAMELIST[0,2];
      COLUMN:=5;
      'FOR' NLPTR:=1 'STEP' 1 'UNTIL' SIZE 'DO'
      'BEGIN'
         'COMMENT' *** work through namelist;
         INC(COLUMN);
         'IF' COLUMN>5 'THEN'
         'BEGIN'
            'COMMENT' *** start in column 1;
            COLUMN:=1;
            ENDLINE(TTO,1);
            INC(LINES);
            'IF' PAUSE=TRUE 'AND' LINES=23 'THEN'
            'BEGIN'
               'COMMENT' *** check continue option if pausing and page full;
               LINES:=1;
               'IF' OPTION(CONTINUE)=FALSE
               'THEN' STOP:=TRUE
```

```
            'END'
          'END';
          'COMMENT' *** escape if discontinuing;
          'IF' STOP=TRUE 'THEN' NLPTR:=SIZE
          'ELSE'
          'BEGIN'
             'COMMENT' *** otherwise output name;
             NAME(TTO,NAMELIST,NLPTR);
             'IF' COLUMN<5 'AND' NLPTR<SIZE 'THEN' NAME TAB(TTO,NAMELIST,NLPTR,17)
          'END'
       'END';
       'IF' STOP=FALSE 'THEN'
       'BEGIN'
          'COMMENT' *** if appropriate send count;
          ENDLINE(TTO,2);
          WRITE NUM(TTO,SIZE,3,0);
          'IF' LIST TYPE=0
          'THEN'
         'BEGIN'
 OUTPUT TEXT(TTO,' SUBSYSTEMS');
        ENDLINE(TTO,1);
        'END'
        'ELSE'
        'BEGIN'
OUTPUT TEXT(TTO,' DATA ITEMS');
        ENDLINE(TTO,1);
        'END'
      'END'
   'END' LISNAM;


   'PROCEDURE' CHNAM ('BYTE''ARRAY' NAMELIST);
   'BEGIN'

      'COMMENT'                                                                    CHNAM (LEVEL 3)
      ***   Changes the name of a subsystem, data item or units identifier.
      ***   The old name is requested and its code found, if it exists.
      ***   Then the new name is requested and a check made to find whether
      ***   the new name is one already existing on the appropriate namelist.
      ***   If the new name doesnt already exist then the text string for
      ***   the new name is simply moved to the namelist to overwrite the
      ***   old name string.  If, however, the new name is one already
      ***   existing then the CONVNM procedure is used to change the
      ***   appropriate REFLIST occurrences of the old name code to the new
      ***   code and remove the old name.  A subsystem name cannot be
      ***   changed to another existing one without confirmation.  If the
      ***   new name is specified as the same as the old then nothing is
      ***   changed.
      ***   Called by: CHSSN          CHDNM
      ***              CHUNIG
      ***   Calls:     OPTION         CONVNM
      ***              FINNAM         MOVSTR;

      'INTEGER' LIST TYPE, OLD CODE, NEW CODE;

      'COMMENT'
      ***   LIST TYPE: Buffer for namelist type.
      ***   OLD CODE: The code of the old name.
      ***   NEW CODE: The code of the new name if it exists;

      LIST TYPE:=NAMELIST[0,2];
           ENDLINE(TTO,1);
```

```
    OUTPUT TEXT(TTO,'OLD NAME .... ');
    INPUT TEXT(TTI);
    FINNAM(NAMELIST,OLD CODE);
    'IF' OLD CODE=0 'THEN' 'BEGIN'
         ENDLINE(TTO,1);
         OUTPUT TEXT(TTO,'NAME DOESNT EXIST');
         ENDLINE(TTO,1);
    'END'
    'ELSE'
    'BEGIN'
       'COMMENT' *** old name code found;
        ENDLINE(TTO,1);
       OUTPUT TEXT(TTO,'NEW NAME .... ');
       INPUT TEXT(TTI);
       FINNAM(NAMELIST,NEW CODE);
       'IF' NEW CODE=0 'THEN'
       'BEGIN'
          'COMMENT' *** new name doesnt already exist;
          MOVSTR(NAMELIST,OLD CODE);
        ENDLINE(TTO,1);
         OUTPUT TEXT(TTO,'  NAME CHANGED');
        ENDLINE(TTO,1);
       'END'
       'ELSE''IF' NEW CODE<>OLD CODE 'THEN'
       'BEGIN'
          'COMMENT' *** new name already exists;
          'IF' LIST TYPE=0 'THEN'
          'BEGIN'
        ENDLINE(TTO,1);
            OUTPUT TEXT(TTO,'SUBSYSTEM NAME ALREADY EXISTS');
            'IF' OPTION('TO GO AHEAD WITH CHANGE')=TRUE
            'THEN' CONVNM(SS NAME,OLD CODE,NEW CODE)
          'END'
          'ELSE' CONVNM(NAMELIST,OLD CODE,NEW CODE)
       'END'
       'ELSE' DO NOTHING; (*** when new and old names are the same)
    'END'
'END' CHNAM;


'PROCEDURE' CHNUME ('VALUE''INTEGER' FIELD);
'BEGIN'

    'COMMENT'                                                                    CHNUME (LEVEL 3)
    ***   Changes rate or precision value of a specific REFLIST entry.
    ***   FIELD specifies rate or prec change.
    ***   Called by: CHRATE       CHPRCE
    ***   Calls:     GETNV        FINENT;

    'INTEGER' LPOS, NEW, MAX, DUM1, DUM2;

    'COMMENT'
    ***   LPOS: Pointer to specific entry.
    ***   NEW: New value of rate or prec.
    ***   DUM1: Subsystem code delivered by FINENT - not used.
    ***   DUM2: Data code delivered by FINENT - also not used;

    FINENT(LPOS,DUM1,DUM2);
    'IF' LPOS>=0 'THEN'
    'BEGIN'
       'COMMENT' *** entry located - proceed with change;
       GETNV(FIELD,NEW);
```

```
        'IF' FIELD=0 'THEN' RATE[LPOS]:=NEW 'ELSE' PREC[LPOS]:=NEW;
          ENDLINE(TTO,1);
        OUTPUT TEXT(TTO,'VALUE CHANGED');
          ENDLINE(TTO,1);
    'END'
'END' CHNUME;


'PROCEDURE' GENNCH ('VALUE''INTEGER' FIELD);
'BEGIN'

    'COMMENT'                                                          GENNCH (LEVEL 3)
    ***   Causes all rates or precisions of a particular value to be changed
    ***   to another specified value throughout the REFLIST.  The change is
    ***   made subsystem by subsystem.  FIELD specifies rate or prec change.
    ***   Called by: CHRATG     CHPRCG
    ***   Calls:     GETNV      GETOLV
    ***              CHNUM;

    'INTEGER' SSCODE, SIZE, OLD, NEW;

    'COMMENT'
    ***   SSCODE: The current subsystem name code.
    ***   SIZE: Buffer for namelist size.
    ***   OLD: Rate or prec value to be changed.
    ***   NEW: New value of rate or prec;

    GETOLV(FIELD,OLD);
    GETNV(FIELD,NEW);
    SIZE:=SS NAME[0,0];
    'FOR' SSCODE:=1 'STEP' 1 'UNTIL' SIZE 'DO' CHNUM(SSCODE,FIELD,OLD,NEW);
      ENDLINE(TTO,1);
    OUTPUT TEXT(TTO,'MOD COMPLETE');
      ENDLINE(TTO,1);
'END' GENNCH;


'PROCEDURE' CHDAV ('VALUE''INTEGER' FIELD);
'BEGIN'

    'COMMENT'                                                          CHDAV (LEVEL 3)
    ***   Sets all rates or precisions associated with a particular data
    ***   item to the specified value, whatever they were before.  FIELD
    ***   specifies rate or precision change.
    ***   Called by: CHDART          CHDAPR
    ***   Calls:     GETNV           LOCNAM;

    'INTEGER' DACODE, NEW, REFPTR;

    'COMMENT'
    ***   DACODE: Specified data name code.
    ***   NEW: Desired rate or prec value.
    ***   REFPTR: Pointer to the REFLISENT;

    LOCNAM('DATA',DATA NAME,DACODE);
    'IF' DACODE>0 'THEN'
    'BEGIN'
      'COMMENT' *** data name exists - proceed with change;
      GETNV(FIELD,NEW);
      'COMMENT' *** search REFLIST for data name occurrence;
      'FOR' REFPTR:=0 'STEP' 1 'UNTIL' ENTRIES-1 'DO'
      'IF' DATA[REFPTR]=DACODE 'THEN'
```

```
      'BEGIN'
         'COMMENT' *** REFLIST entry involves specified data item;
         'IF' FIELD=0 'THEN' RATE[REFPTR]:=NEW 'ELSE' PREC[REFPTR]:=NEW
      'END';
        ENDLINE(TTO,1);
      OUTPUT TEXT(TTO,'MOD COMPLETE');
        ENDLINE(TTO,1);
   'END'
'END' CHDAV;


'PROCEDURE' CHSSV ('VALUE''INTEGER' FIELD);
'BEGIN'

   'COMMENT'                                                                    CHSSV (LEVEL 3)
   ***   Invokes the change of all rates or precisions of a particular
   ***   value to another specified value for relevant REFLIST entries
   ***   relating to the subsystem specified by the user.  FIELD specifies
   ***   rate or precision change.
   ***   Called by: CHSSRT          CHSSPR
   ***   Calls:     GETNV           GETOLV
   ***              CHNUM           LOCNAM;

   'INTEGER' OLD, NEW, SSCODE;

   'COMMENT'
   ***   OLD: Rate or prec value to be changed.
   ***   NEW: New value of rate or prec.
   ***   SSCODE: Code of specified subsystem;

   LOCNAM('SUBSYSTEM',SS NAME,SSCODE);
   'IF' SSCODE>0 'THEN'
   'BEGIN'
      'COMMENT' *** subsystem exists - invoke change;
      GETOLV(FIELD,OLD);
      GETNV(FIELD,NEW);
      CHNUM(SSCODE,FIELD,OLD,NEW);
        ENDLINE(TTO,1);
      OUTPUT TEXT(TTO,'MOD COMPLETE');
        ENDLINE(TTO,1);
   'END'
'END' CHSSV;


'PROCEDURE' DELENT ('VALUE''INTEGER' POS);
'BEGIN'

   'COMMENT'                                                                  DELENT (LEVEL 3)
   ***   Removes the entry at position POS from the REFLIST database
   ***   by overwriting.  All higher entries are moved down one place.
   ***   Called by: DELDAR   DEL1EN;

   'INTEGER' REFPTR;

   'COMMENT'
   ***   REFPTR: Pointer to the REFLISENT;

   'FOR' REFPTR := POS 'STEP' 1 'UNTIL' ENTRIES - 1 'DO'
   'BEGIN'
      SS[REFPTR] := SS[REFPTR+1];
      DATA[REFPTR] := DATA[REFPTR+1];
      TR[REFPTR] := TR[REFPTR+1];
```

```
      RATE[REFPTR] := RATE[REFPTR+1];
      PREC[REFPTR] := PREC[REFPTR+1];
      UNITS[REFPTR] := UNITS[REFPTR+1]
   'END';
   DEC (ENTRIES)
'END' DELENT;

'PROCEDURE' FILREF ('VALUE''INTEGER' CODE);
'BEGIN'

   'COMMENT'                                                                    FILREF (LEVEL 3)
   ***   Sends either complete or partial REFLIST to a disc file, the name
   ***   of which is supplied by FINFIL.  If CODE is 0 then the complete
   ***   REFLIST is sent, otherwise only those entries relating to the
   ***   subsystem whose code is CODE are sent.  The output is preceded with
   ***   the relevant count of entries.  In filing the complete REFLIST,
   ***   entries are sent subsystem by subsystem.
   ***   Called by: FILENT            FILESD
   ***   Calls:     FINFIL            ACTPRC
   ***              FILNUM;

   'INTEGER' REFPTR, ENTRYCOUNT, SSCODE, LOWER, UPPER, ERROR;

   'COMMENT'
   ***   REFPTR: Pointer to the REFLIST entries.
   ***   ENTRYCOUNT: Count of entries to be sent.
   ***   SSCODE: The current subsystem name code.
   ***   LOWER: Lower bound subsystem code for output loop.
   ***   UPPER: Upper bound subsystem code for output loop.
   ***   ERROR: Dummy error flag for FINFIL;

   FINFIL("DATA",1, ERROR);
     OUTPUT TEXT(OUTFIL, "REFLIST FILE");
     ENDLINE(OUTFIL,1);
   'IF' CODE<>0 'THEN'
   'BEGIN'
      'COMMENT' *** find how many entries relate to specific subsystem;
      ENTRYCOUNT:=0;
      'FOR' REFPTR:=0 'STEP' 1 'UNTIL' ENTRIES-1 'DO' 'IF' SS[REFPTR]=CODE 'THEN' INC(ENTRYCOUNT)
   'END'
   'ELSE' ENTRYCOUNT:=ENTRIES;
   FILNUM(ENTRYCOUNT);
   'COMMENT' *** set lower and upper bound subsystems for output loop;
   LOWER:='IF' CODE=0 'THEN' 1 'ELSE' CODE;
   UPPER:='IF' CODE=0 'THEN' SS NAME[0,0] 'ELSE' CODE;
   'COMMENT' *** for each subsystem in turn, search REFLIST for relevant entries;
   'FOR' SSCODE:=LOWER 'STEP' 1 'UNTIL' UPPER 'DO'
   'FOR' REFPTR:=0 'STEP' 1 'UNTIL' ENTRIES-1 'DO'
   'IF' SS[REFPTR]=SSCODE 'THEN'
   'BEGIN'
      'COMMENT' *** REFLIST entry refers to desired subsystem - output details;
      NAME(OUTFIL,SS NAME,SS[REFPTR]); ENDLINE(OUTFIL,1);
      NAME(OUTFIL,DATA NAME,DATA[REFPTR]); ENDLINE(OUTFIL,1);
      'IF' TR[REFPTR]=1 'THEN' OUTPUT TEXT(OUTFIL,"T") 'ELSE' OUTPUT TEXT(OUTFIL,"R");
      ENDLINE(OUTFIL,1);
      FILNUM(RATE[REFPTR]);
      FILNUM(ACTPRC(PREC[REFPTR]));
      NAME(OUTFIL,UNITS NAME,UNITS[REFPTR]); ENDLINE(OUTFIL,1)
   'END';
   CLOSE WRITE FILE
'END' FILREF;
```

FRMTXS (LEVEL 3)

```
'PROCEDURE' FRMTXS;
'BEGIN'

   'COMMENT'
   ***   During configuration of a system, this procedure determines which
   ***   subsystems are to be included, moving the appropriate codes onto
   ***   the CONF SS LIST and initialising the corresponding TERM ADDRESS
   ***   entries.  The user is first offered the option to include all
   ***   subsystems.  If this is declined, each subsystem in turn is offered
   ***   and the user indicates whether or not it is to be included.  If the
   ***   user declines all subsystems the fact is reported.
   ***   Called by: CONFIG
   ***   Calls:     OPTION;

   'INTEGER' SSPTR, SIZE;

   'COMMENT'
   ***   SSPTR: Pointer to SS NAME entries.
   ***   SIZE: Buffer for namelist size;
   SIZE:=SS NAME[0,0];
   'IF' OPTION("TO INCLUDE ALL SUBSYSTEMS")=TRUE 'THEN'
   'BEGIN'
      'COMMENT' *** all subsystems included;
      CONF SS COUNT:=SIZE;
      'FOR' SSPTR:=1 'STEP' 1 'UNTIL' SIZE 'DO'
         'BEGIN'
         'COMMENT' *** transfer all name codes;
            CONF SS LIST[SSPTR]:=SSPTR;
            TERM ADDRESS[SSPTR]:=-1;
         'END'
      'END'
   'ELSE'
   'BEGIN'
      'COMMENT' *** not all subsystems included;
      CONF SS COUNT:=0;
      'FOR' SSPTR:=1 'STEP' 1 'UNTIL' SIZE 'DO'
        'BEGIN'
         'COMMENT' *** offer each name in turn;
          ENDLINE(TTO,1);
          NAME(TTO,SS NAME,SSPTR);
          'IF' OPTION("TO INCLUDE")=TRUE 'THEN'
             'BEGIN'
            'COMMENT' *** this subsystem included - copy name code;
             INC(CONF SS COUNT);
             CONF SS LIST[CONF SS COUNT]:=SSPTR;
             TERM ADDRESS[CONF SS COUNT]:=-1
             'END'
        'END'
       'END';
    'IF' CONF SS COUNT=0 'THEN'
       'BEGIN'
       ENDLINE(TTO,1);
       OUTPUT TEXT(TTO,"NO SYSTEM CONFIGURED");
       ENDLINE(TTO,1);
       'END'
'END' FRMTXS;


'PROCEDURE' FRMTXD;
'BEGIN'
```

```
    'COMMENT'                                                          FRMTXD (LEVEL 3)
    ***   During the configuration of a system, this procedure searches the
    ***   REFLIST to locate all data items transmitted by each configured
    ***   subsystem in turn.  Each item thus found is recorded in the next
    ***   available TXDATA element of TXLIST, with associated subsystem,
    ***   rate, precision and units values being set up in the appropriate
    ***   fields of the entry.
    ***   Called by: CONFIG;

    'INTEGER' CSSPTR, REFPTR, SSCODE;

    'COMMENT'
    ***   CSSPTR: Pointer to configured subsystem entries.
    ***   REFPTR: Pointer to the REFLIST entries.
    ***   SSCODE: The current subsystem name code;

    TOTAL TX DATA := 0;
    'FOR' CSSPTR := 1 'STEP' 1 'UNTIL' CONF SS COUNT 'DO'
    'BEGIN'
       'COMMENT' *** take each configured subsystem in turn;
       SSCODE := CONF SS LIST[CSSPTR];
       'FOR' REFPTR := 0 'STEP' 1 'UNTIL' ENTRIES - 1 'DO'
       'BEGIN'
          'COMMENT' *** search REFLIST entries;
          'IF' SS[REFPTR] = SSCODE 'AND' TR[REFPTR] = 1 'THEN'
          'BEGIN'
             'COMMENT' *** located data item transmitted by the subsystem;
             TXSS[TOTAL TX DATA] := SSCODE;
             TXDATA[TOTAL TX DATA] := DATA[REFPTR];
             TXRATE[TOTAL TX DATA] := RATE[REFPTR];
             TXPREC[TOTAL TX DATA] := PREC[REFPTR];
             TXUNITS[TOTAL TX DATA] := UNITS[REFPTR];
             NRX[TOTAL TX DATA] := 0;
             CONF[REFPTR] := TRUE;
             INC (TOTAL TX DATA)
          'END'
       'END'
    'END';
'END' FRMTXD;
'PROCEDURE' FRMRXS;
'BEGIN'

    'COMMENT'                                                          FRMRXS (LEVEL 3)
    ***   During the configuration of a system, this procedure searches the
    ***   REFLIST in order to locate all configured receivers of each
    ***   transmitted data item in TXLIST, placing their codes in a sublist
    ***   of the RXUNIT elements of the RXLIST and noting their rates,
    ***   precisions and units in the appropriate RXRATE, RXPREC and
    ***   RXUNITS elements.  Places a pointer to the start of the sublist
    ***   in the RXSTART element of TXLIST for this data item and records
    ***   the size of the sublist, ie the number of receivers for the item,
    ***   in the corresponding NRX element.  For each entry thus dealt with,
    ***   the fact is noted in the CONF element of REFLIST.
    ***   Called by: CONFIG
    ***   Calls:     WORDS                    ACTPRC;

    'INTEGER' TXPTR, REFPTR, RXPTR, CSSPTR, DACODE, CFLAG, RXCOUNT;

    'COMMENT'
    ***   TXPTR: Pointer to TXLIST entries.
    ***   REFPTR: Pointer to the REFLIST entries.
    ***   RXPTR: Pointer to RXLIST entries.
```

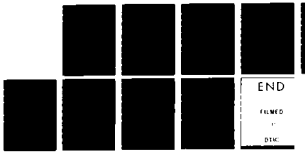
```
    ***   CSSPTR: Pointer to confisured subsystem entries.
    ***   DACODE: Code of the current transmitted data item.
    ***   CFLAG: Flas to indicate if located receiver is confisured.
    ***   RXCOUNT: Runnins count of entries on RXLIST;

    TXPTR := 0; RXCOUNT := 0;
    WHILE (TXPTR < TOTAL TX DATA) 'DO'
    'BEGIN'
       'COMMENT' *** take each transmitted data item in turn;
       DACODE := TXDATA[TXPTR];
       REFPTR := 0;
       WHILE (REFPTR < ENTRIES) 'DO'
       'BEGIN'
          'COMMENT' *** search REFLIST for receive entries of data item;
          'IF' DATA[REFPTR] = DACODE 'AND' TR[REFPTR] = 0 'THEN'
          'BEGIN'
             'COMMENT' *** located a receive entry - check if receiver confisured;
             CFLAG := FALSE;
             'FOR' CSSPTR := 1 'STEP' 1 'UNTIL' CONF SS COUNT 'DO'
             'IF' SS[REFPTR] = CONF SS LIST[CSSPTR] 'THEN' CFLAG := TRUE;
             'IF' CFLAG = TRUE 'THEN'
             'BEGIN'
                'COMMENT' *** receiver confisured - update TXLIST and RXLIST;
                RXSTART[TXPTR] := RXCOUNT;
                RXPTR := RXCOUNT + NRX[TXPTR];
                RXSS[RXPTR] := SS[REFPTR];
                RXRATE[RXPTR] := RATE[REFPTR];
                RXPREC[RXPTR] := PREC[REFPTR];
                RXUNITS[RXPTR] := UNITS[REFPTR];
                RXWORDS[RXPTR] := WORDS (ACTPRC (PREC[REFPTR]));
                INC (NRX[TXPTR]);
                CONF[REFPTR] := TRUE
             'END'
          'END';
          INC (REFPTR)
       'END';
       RXCOUNT := RXCOUNT + NRX[TXPTR];
       INC (TXPTR)
    'END'
 'END' FRMRXS;
'PROCEDURE' FATLCK ('LOCATION''INTEGER' FATAL);
'BEGIN'

    'COMMENT'                                                          FATLCK (LEVEL 3)
    ***   Checks for fatal inconsistencies in a confisured system.
    ***   Checks whether the samedata is transmitted by more than one
    ***   subsystem, if a data item is received by the subsystem which
    ***   transmits it, and whether any rate, precision or units
    ***   requirement is incompatible with that transmitted.
    ***   If any such inconsistencies exist, the facts are reported and
    ***   FATAL returned TRUE.
    ***   Called by: CONFIG        CKCONS
    ***   Calls:     FATLPR        ACTPRC;

    'INTEGER' TXPTR1, TXPTR2, NRXPTR, HEADER, RXPTR;

    'COMMENT'
    ***   TXPTR1: Pointer to TXLIST entries.
    ***   TXPTR2: Pointer to TXLIST entries for multiple tx check.
    ***   DATARX: Index to current receiver of data item beins examined.
    ***   HEADER: Flas to indicate header needed on multiple tx output list.
    ***   RXPTR: Pointer to RXLIST entries;
```

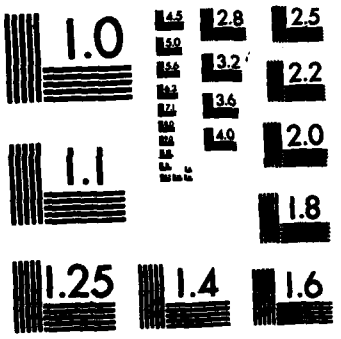
```
'COMMENT' *** first initialise TXCHECK entries;
'FOR' TXPTR1:=0 'STEP' 1 'UNTIL' TOTAL TX DATA-1 'DO' TXCHECK[TXPTR1]:=0;
'FOR' TXPTR1:=0 'STEP' 1 'UNTIL' TOTAL TX DATA-1 'DO'
'BEGIN'
   'COMMENT' *** take each transmitted data item in turn;
   HEADER:=TRUE;
   TXPTR2:=TXPTR1+1;
   WHILE (TXPTR2<TOTAL TX DATA) 'DO'
   'BEGIN'
      'COMMENT' *** look for another transmitter of same data;
     'IF' TXDATA[TXPTR1]=TXDATA[TXPTR2] 'AND' TXCHECK[TXPTR1]=0 'THEN'
      'BEGIN'
         'COMMENT' *** we've found one not already noted;
         'IF' HEADER=TRUE 'THEN'
         'BEGIN'
            'COMMENT' *** identify data and first transmitter;
             ENDLINE(TTO,1);
            OUTPUT TEXT(TTO,'*** ');
            NAME(TTO,DATA NAME,TXDATA[TXPTR1]);
            OUTPUT TEXT(TTO,' TRANSMITTED BY ');
            NAME(TTO,SS NAME,TXSS[TXPTR1]);
            ENDLINE(TTO,1);
            HEADER:=FALSE
         'END';
         'COMMENT' *** identify other transmitter;
         OUTPUT TEXT(TTO,'                    AND ');
         NAME(TTO,SS NAME,TXSS[TXPTR2]);
         ENDLINE(TTO,1);
         TXCHECK[TXPTR2]:=1;
         FATAL:=TRUE
      'END';
      INC(TXPTR2)
   'END';
   'COMMENT' *** now check receiver requirements;
   NRXPTR:=0;
   WHILE (NRXPTR<NRX[TXPTR1]) 'DO'
   'BEGIN'
      'COMMENT' *** look at each receiver in turn;
      RXPTR:=RXSTART[TXPTR1]+NRXPTR;
      'IF' RXSS[RXPTR]=TXSS[TXPTR1] 'THEN'
      'BEGIN'
         'COMMENT' *** this receiver is the same subsystem as transmitter;
         ENDLINE(TTO,1);
         OUTPUT TEXT(TTO,'*** ');
         NAME(TTO,DATA NAME ,TXDATA[TXPTR1]);
         OUTPUT TEXT(TTO,' TRANSMITTED AND RECEIVED BY ');
         NAME(TTO,SS NAME ,TXSS[TXPTR1]);
         ENDLINE(TTO,1);
         FATAL:=TRUE
      'END';
      'COMMENT'  *** check incompatible rates, precisions and units;
      'IF' RXRATE[RXPTR]>0 'THEN'
      'BEGIN'
         'IF' TXRATE[TXPTR1]=0 'OR' RXRATE[RXPTR]<TXRATE[TXPTR1]
         'THEN' FATLPR(0,TXPTR1,RXPTR,FATAL)
      'END';
      'IF' ACTPRC(RXPREC[RXPTR])>ACTPRC(TXPREC[TXPTR1])
      'THEN' FATLPR(1,TXPTR1,RXPTR,FATAL);
      'IF' RXUNITS[RXPTR]<>TXUNITS[TXPTR1] 'THEN' FATLPR(2,TXPTR1,RXPTR,FATAL);
      INC(NRXPTR)
   'END'
```

```
    'END';
    'IF' FATAL=FALSE 'THEN' 'BEGIN'
OUTPUT TEXT(TTO,'NO FATAL INCONSISTENCIES'); ENDLINE(TTO,1)
  'END'
  'END' FATLCK;


  'PROCEDURE' FRMMES;
  'BEGIN'

    'COMMENT'                                                                    FRMMES (LEVEL 3)
    ***   During the configuration of a system, this procedure forms the
    ***   bus messages, creating the MESSAGE HEADER table entries and
    ***   assembling the associated data word information in the MESSAGE
    ***   DATA array.  The formation of messages takes place in order
    ***   of rate, with Rate 0 transfers dealt with first, then Rate 1
    ***   and so on.  For each rate, the messages are formed in order
    ***   of source subsystem and within this in order of sink subsystem.
    ***   The rate, source and sink having been set up, the TXLIST table is
    ***   searched for a data item with the correct source.  For each entry
    ***   thus found, the appropriate RXLIST entries are examined to
    ***   find a receiving subsystem which matches the sink and with the
    ***   desired rate requirement.  The first complete match discovered
    ***   for this rate, source and sink causes the next entry of the
    ***   MESSAGE HEADER table to be set up and initialises the pointer
    ***   to the MESSAGE DATA array entries.  The data item is then
    ***   placed in the first element of the corresponding row of the
    ***   MESSAGE DATA array.  Subsequent matches cause the appropriate
    ***   data codes to be recorded in the following elements of the
    ***   MESSAGE DATA array row.  If more than one word is required to
    ***   represent a data item, as discovered from the RXWORDS field
    ***   of RXLIST, then the RXWORDS field is decremented and the
    ***   pointer to TXLIST is decremented, so that it will point to
    ***   the same TXLIST entry again, instead of moving on to examine the
    ***   next.  This means that the same data item is taken account of
    ***   as many times as necessary to build up the correct number of
    ***   entries in the MESSAGE DATA array.  If at any time the number of
    ***   words created for the message equals the value of MESSAGE
    ***   LENGTH, then the message information is completed by inserting
    ***   the correct value in the WDS field of the MESSAGE HEADER entry,
    ***   and the next data item found which matches the criteria, if
    ***   any, will cause a new message entry to be created.  When all
    ***   data words have been discovered for the target rate, source and
    ***   sink, the message entry being filled at the time is completed
    ***   by having its WDS field set.  The search then continues for the
    ***   next sink, source or rate.
    ***   Called by: CONFIG;

    'INTEGER' RGRP, TXCODE, RXCODE, HEADER, DITEM, DATARX, RXPTR, MCPTR, WCNT;

    'COMMENT'
    ***   RGRP: The current value of rate group.
    ***   TXCODE: The position in the CONF SS LIST of the current source.
    ***   RXCODE: The position in the CONF SS LIST of the current sink.
    ***   HEADER: Flag to indicate MESSAGE HEADER needs to be set up.
    ***   TXPTR: Pointer to TXLIST entries.
    ***   DATARX: Index to current receiver of data item being examined.
    ***   RXPTR: Pointer to RXLIST entries.
    ***   MDPTR: Pointer to data words in message data array.
    ***   WCNT: Buffer for RXWORDS value;

    TOTAL MESSAGES:=0;
```

AD-A124 608 MODIFICATION OF SAVANT TO RUN ON THE PDP RSX-11M OPERATING SYSTEM(U) ROYAL AIRCRAFT ESTABLISHMENT FARNBOROUGH (ENGLAND) R J BLUFF ET AL. JAN 82 2/2

UNCLASSIFIED RAE-TM-FS(F)452 DRIC-BR-85532 F/G 9/2 NL





MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS-1963-A

```
  ENDLINE(TTO,1);
OUTPUT TEXT(TTO,'THIS MIGHT TAKE SOME TIME.....');
'FOR' RGRP:=0 'STEP' 1 'UNTIL' LOWEST RATE 'DO'
'FOR' TXCODE:=1 'STEP' 1 'UNTIL' CONF SS COUNT 'DO'
'FOR' RXCODE:=1 'STEP' 1 'UNTIL' CONF SS COUNT 'DO'
'IF' TXCODE<>RXCODE 'THEN'
'BEGIN'
   'COMMENT' *** rate and source/sink pair set up;
   HEADER:=TRUE;
   DITEM:=0;
   WHILE (DITEM<TOTAL TX DATA) 'DO'
   'BEGIN'
      'COMMENT' *** work through TXLIST;
      'IF' TXSS[DITEM]=CONF SS LIST[TXCODE] 'THEN'
      'BEGIN'
         'COMMENT' *** found message where source matches;
         DATARX:=0;
         WHILE (DATARX<NRX[DITEM]) 'DO'
         'BEGIN'
            'COMMENT' *** investigate each receiver in turn;
            RXPTR:=RXSTART[DITEM]+DATARX;
            'IF' RXRATE[RXPTR]=RGRP 'AND' RXSS[RXPTR]=CONF SS LIST[RXCODE] 'THEN'
            'BEGIN'
               'COMMENT' *** sink and rate match;
               'IF' HEADER=TRUE 'THEN'
               'BEGIN'
                  'COMMENT' *** set up message header entries;
                  TX[TOTAL MESSAGES]:=CONF SS LIST[TXCODE];
                  RX[TOTAL MESSAGES]:=CONF SS LIST[RXCODE];
                  RG[TOTAL MESSAGES]:=RGRP;
                  RETRY[TOTAL MESSAGES]:=0;
                  WDS[TOTAL MESSAGES]:=0;
                  MCPTR:=1;
                  HEADER:=FALSE
               'END';
               MESSAGE DATA[TOTAL MESSAGES,MCPTR]:=TXDATA[DITEM];
               (*** place data code in MESSAGE DATA array)
               INC(MCPTR);
               'IF' MCPTR>MESSAGE LENGTH 'THEN'
               'BEGIN'
                  'COMMENT' *** message is full;
                  WDS[TOTAL MESSAGES]:=MESSAGE LENGTH;
                  INC(TOTAL MESSAGES);
                  HEADER:=TRUE
               'END';
               'COMMENT' *** see if more words needed for the data item;
               WCNT:=RXWORDS[RXPTR];
               'IF' WCNT=0 'THEN' WCNT:=64;
               DEC(WCNT);
               RXWORDS[RXPTR]:=WCNT;
               DATARX:=NRX[DITEM];
               'IF' WCNT>0 'THEN' DEC(DITEM)
            'END';
            INC(DATARX)
         'END'
      'END';
      INC(DITEM)
   'END';
   'IF' WDS[TOTAL MESSAGES]=0 'AND' HEADER=FALSE 'THEN'
   'BEGIN'
      'COMMENT' *** round off message with word count;
      WDS[TOTAL MESSAGES]:=MCPTR-1;
```

```
        INC(TOTAL MESSAGES)
      'END'
   'END'
'END' FRMMES;


'PROCEDURE' FINBSS ('LOCATION''INTEGER' FATAL);
'BEGIN'

   'COMMENT'                                                          FINBSS (LEVEL 3)
   ***   After forming messages, locates all subsystems which are
   ***   on the data bus, marking them as such in the TERM ADDRESS
   ***   array.  The search takes place in order of configured subsystem,
   ***   with the message list being examined, either until a message
   ***   is found which involves the subsystem in a transfer with
   ***   a rate group value >0 or until the list is exhausted.
   ***   If the first case holds, then the appropriate entry of TERM
   ***   ADDRESS is marked with a 0, otherwise it is left with the
   ***   value of -1 set up when the array was initialised.  A running
   ***   count of bus terminals is kept, and if this exceeds TERMINAL
   ***   LIMIT then a message is displayed and the parameter FATAL is
   ***   returned TRUE to the calling procedure.
   ***   Called by: CONFIG;

   'INTEGER' SSCODE, CSSPTR, MPTR, BUSCOUNT;

   'COMMENT'
   ***   SSCODE: The current subsystem name code.
   ***   CSSPTR: Pointer to configured subsystem entries.
   ***   MPTR: Pointer to message entries.
   ***   BUSCOUNT: The count of bus-connected subsystems;

   BUSCOUNT:=0;
   'FOR' CSSPTR:=1 'STEP' 1 'UNTIL' CONF SS COUNT 'DO'
   'BEGIN'
      'COMMENT' *** work through each subsystem in turn;
      SSCODE:=CONF SS LIST[CSSPTR];
      MPTR:=0;
      WHILE (MPTR<TOTAL MESSAGES) 'DO'
      'BEGIN'
         'COMMENT' *** look for messages involving this subsystem;
         'IF' TX[MPTR]=SSCODE 'OR' RX[MPTR]=SSCODE 'THEN'
         'BEGIN'
            'COMMENT' *** found relevant message;
            'IF' RG[MPTR]<>0 'THEN'
            'BEGIN'
               'COMMENT' *** and rate value is >0;
               TERM ADDRESS[CSSPTR]:=0;
               INC(BUSCOUNT);
               MPTR:=TOTAL MESSAGES; (*** search no more)
            'END'
         'END';
         INC(MPTR)
      'END'
   'END';
   'IF' BUSCOUNT>TERMINAL LIMIT 'THEN'
   'BEGIN'
      'COMMENT' *** too many bus subsystems;
     ENDLINE(TTO,1);
      OUTPUT TEXT(TTO,'*** FATAL:');
     ENDLINE(TTO,1);
      OUTPUT TEXT(TTO,'THIS SYSTEM CONTAINS MORE THAN');
```

```
      WRITE NUM(TTO,TERMINAL LIMIT,2,0);
     ENDLINE(TTO,1);
      OUTPUT TEXT(TTO,' TERMINALS AND IS THEREFORE INVALID FOR FURTHER ANALYSIS');
      FATAL:=TRUE
   'END'
'END' FINDSS;


'PROCEDURE' PARNUM;
'BEGIN'

   'COMMENT'                                                                    PARNUM (LEVEL 3)
   $$$   After the formation of messages, places the partition count with
   $$$   data items which have been partitioned.  It does this by
   $$$   examining the contents of each message in turn.  If the current
   $$$   data word is the same as the previous one in the message, then
   $$$   the count of partitions is incremented and placed with the
   $$$   current word.  If the previous word was not itself marked as
   $$$   being partitioned, determined with reference to the COUNTING
   $$$   flag, then this causes the flag to be set, the previous word
   $$$   to be marked, and the partition count to be initialised.  If
   $$$   the current word is not the same as the previous, and
   $$$   partitioning has been taking place up to now, then this causes
   $$$   the COUNTING flag to be reset.  The case where partitioning
   $$$   may extend over a word boundary has to be taken into account,
   $$$   and for this purpose the 0'th element of the MESSAGE DATA
   $$$   array row is utilised.  If the current message being examined
   $$$   has the same parameters as the previous, meaning that this
   $$$   message is a continuation, then the last word in the previous
   $$$   message is transferred into the 0'th word of the current, in
   $$$   order that the comparison algorithm can operate throughout
   $$$   the message.  The only other check that has to be done is that
   $$$   if a data partition has been discovered in the first data word
   $$$   of the current message, by comparison with the last word of
   $$$   the previous, then the count value 1 has to be implanted in
   $$$   the partition count of the previous message's last word.
   $$$   Called by: CONFIG;

   'INTEGER' MPTR, MDPTR, COUNTING, COUNT;

   'COMMENT'
   $$$   MPTR: Pointer to message entries.
   $$$   MDPTR: Pointer to data words in message data array.
   $$$   COUNTING: Flag to indicate partitioning is taking place.
   $$$   COUNT: Value of the partition count;

   COUNTING := FALSE;
   'FOR' MPTR := 0 'STEP' 1 'UNTIL' TOTAL MESSAGES - 1 'DO'
   'BEGIN'
      'COMMENT'  $$$ work through messages;
      DATAWORD (MPTR, 0) := 'IF' MPTR > 0
                        'AND' RG[MPTR] = RG[MPTR-1]
                        'AND' TX[MPTR] = TX[MPTR-1]
                        'AND' RX[MPTR] = RX[MPTR-1]
                        'THEN' DATAWORD (MPTR-1, MESSAGE LENGTH)
                        'ELSE' 0;
                        ($$$ set up 0'th element if relevant)
      'FOR' MDPTR := 1 'STEP' 1 'UNTIL' WDS[MPTR] 'DO'
      'IF' DATAWORD (MPTR, MDPTR) = DATAWORD (MPTR, MDPTR-1) 'THEN'
      'BEGIN'
         'COMMENT' $$$ found partitioned data word;
         'IF' COUNTING = FALSE 'THEN'
```

```
            'BEGIN'
               'COMMENT' $$$ first encounter with this partitioned word;
               COUNTING := TRUE;
               COUNT := 2; ($$$ initialise count)
               PARTITION COUNT (MPTR, MDPTR-1) := 1; ($$$ plant start of count in previous word)
               'IF' MDPTR = 1 'THEN' PARTITION COUNT (MPTR-1, MESSAGE LENGTH) := 1
            'END';
            PARTITION COUNT (MPTR, MDPTR) := COUNT;
            INC (COUNT)
         'END'
         'ELSE' COUNTING := FALSE
      'END'
   'END' PARNUM;

'PROCEDURE' SETMAP;
'BEGIN'

   'COMMENT'                                                                    SETMAP (LEVEL 3)
   $$$   Sets up the MAP array prior to mapping data traffic.  Places
   $$$   integers to label columns and rows in the appropriate elements,
   $$$   up to the number of configured subsystems in the system, with
   $$$   leading zeroes being replaced by spaces.  Then fills the correct
   $$$   number of elements in the matrix with dot (.) characters, except
   $$$   for the elements of the leading diagonal, which indicate the
   $$$   same source and sink.  These are filled with * characters.
   $$$   Called by: MAPDAT;

   'INTEGER' TENS, UNITS, ROW, COLUMN, INDEX, CHAR;

   'COMMENT'
   $$$   TENS: Tens character of row or column label.
   $$$   UNITS: Units character of row or column label.
   $$$   ROW: Row pointer to element.
   $$$   COLUMN: Column pointer to element.
   $$$   INDEX: Count of rows or columns labelled.
   $$$   CHAR: Internal value of telecode character;

   'COMMENT' $$$ first put tens characters in row and column labels;
   'FOR' TENS := 0 'STEP' 1 'UNTIL' 6 'DO'
   'FOR' UNITS := 0 'STEP' 1 'UNTIL' 9 'DO'
   'BEGIN'
      'COMMENT' $$$ count up to number of configured subsystems;
      INDEX := 10*TENS + UNITS;
      'IF' INDEX <= CONF SS COUNT 'THEN'
      'BEGIN'
         CHAR := 'IF' TENS = 0 'THEN' 160 'ELSE' 176 + TENS;
         MAP[-1, INDEX] := CHAR; ($$$ set up tens character in column label)
         MAP[INDEX, -1] := CHAR; ($$$ set up tens character in row label)
      'END'
   'END';
   'COMMENT' $$$ now put units character in row and column labels;
   'FOR' UNITS := 0 'STEP' 1 'UNTIL' 9 'DO'
   'FOR' TENS := 0 'STEP' 10 'UNTIL' 60 'DO'
   'BEGIN'
      INDEX := TENS + UNITS;
      'IF' INDEX <= CONF SS COUNT 'THEN'
      'BEGIN'
         CHAR := 176 + UNITS;
         MAP[0, INDEX] := CHAR; ($$$ set up units character in column label)
         MAP[INDEX, 0] := CHAR; ($$$ set up units character in row label)
      'END'
   'END';
```

```
    'COMMENT' now put in space, dot and asterisk characters;
    MAP[-1, -1] := 160;
    MAP[0, 0] := 160;
    'FOR' ROW := 1 'STEP' 1 'UNTIL' CONF SS COUNT 'DO'
    'FOR' COLUMN := 1 'STEP' 1 'UNTIL' CONF SS COUNT 'DO'
    MAP[ROW, COLUMN] := 'IF' ROW = COLUMN 'THEN' 170 'ELSE' 174
'END' SETMAP;

'PROCEDURE' LISSSP ('VALUE''INTEGER' TXCODE, RXCODE; 'LOCATION''INTEGER' OK);
'BEGIN'

    'COMMENT'                                                          LISSSP (LEVEL 3)
    $$$   Lists all data items passing from the subsystem whose position
    $$$   on CONF SS LIST is delivered as TXCODE to the subsystem whose
    $$$   position on CONF SS LIST is delivered as RXCODE.  OK is a flag
    $$$   returned to the calling procedure to indicate whether or not
    $$$   any traffic has been found to exist between this source and
    $$$   sink - this is determined by whether or not a header has been
```

APPENDIX F1

SAVANT.ODL

```
        .ROOT   SAVOVL-LB:[1,1]COROTS/LB:CRIWRT:CRSPAC:CRWRCH-LIB2-LIB-*(OVERLY)
OVERLY: .FCTR   (SAV00,SAV01,SAV02,SAV03,SAV04,SAV05,SAV06,SAV07,CONT1)
CONT1:  .FCTR   (SAV08,SAV09,SAV10,SAV11,SAV12,SAV13,SAV14,SAV15,CONT2)
CONT2:  .FCTR   (SAV16,SAV17,SAV18,SAV19,SAV20,SAV21,SAV22,SAV23,CONT3)
CONT3:  .FCTR   (SAV24,SAV25,SAV26,SAV27,SAV29,SAV30,SAV31,CONT4)
CONT4:  .FCTR   (SAV32,SAV33,SAV34,SAV35,SAV36,SAV37,SAV38,SAV39,CONT5)
CONT5:  .FCTR   (SAV40,SAV41,SAV42,SAV43,SAV44,SAV45,SAV46,SAV47,CONT6)
CONT6:  .FCTR   (SAV48,SAV49,SAV50,SAV51)

SAV00:  .FCTR   SAVLIB/LB:NUM00-LB:[32,1]SAVLIB/LB
SAV01:  .FCTR   SAVLIB/LB:NUM01-LB:[32,1]SAVLIB/LB
SAV02:  .FCTR   SAVLIB/LB:NUM02-LB:[32,1]SAVLIB/LB
SAV03:  .FCTR   SAVLIB/LB:NUM03-LB:[32,1]SAVLIB/LB
SAV04:  .FCTR   SAVLIB/LB:NUM04-LB:[32,1]SAVLIB/LB
SAV05:  .FCTR   SAVLIB/LB:NUM05-LB:[32,1]SAVLIB/LB
SAV06:  .FCTR   SAVLIB/LB:NUM06-LB:[32,1]SAVLIB/LB
SAV07:  .FCTR   SAVLIB/LB:NUM07-LB:[32,1]SAVLIB/LB
SAV08:  .FCTR   SAVLIB/LB:NUM08-LB:[32,1]SAVLIB/LB
SAV09:  .FCTR   SAVLIB/LB:NUM09-LB:[32,1]SAVLIB/LB
SAV10:  .FCTR   SAVLIB/LB:NUM10-LB:[32,1]SAVLIB/LB
SAV11:  .FCTR   SAVLIB/LB:NUM11-LB:[32,1]SAVLIB/LB
SAV12:  .FCTR   SAVLIB/LB:NUM12-LB:[32,1]SAVLIB/LB
SAV13:  .FCTR   SAVLIB/LB:NUM13-LB:[32,1]SAVLIB/LB
SAV14:  .FCTR   SAVLIB/LB:NUM14-LB:[32,1]SAVLIB/LB
SAV15:  .FCTR   SAVLIB/LB:NUM15-LB:[32,1]SAVLIB/LB
SAV16:  .FCTR   SAVLIB/LB:NUM16-LB:[32,1]SAVLIB/LB
SAV17:  .FCTR   SAVLIB/LB:NUM17-LB:[32,1]SAVLIB/LB
SAV18:  .FCTR   SAVLIB/LB:NUM18-LB:[32,1]SAVLIB/LB
SAV19:  .FCTR   SAVLIB/LB:NUM19-LB:[32,1]SAVLIB/LB
SAV20:  .FCTR   SAVLIB/LB:NUM20-LB:[32,1]SAVLIB/LB
SAV21:  .FCTR   SAVLIB/LB:NUM21-LB:[32,1]SAVLIB/LB
SAV22:  .FCTR   SAVLIB/LB:NUM22-LB:[32,1]SAVLIB/LB
SAV23:  .FCTR   SAVLIB/LB:NUM23-LB:[32,1]SAVLIB/LB
SAV24:  .FCTR   SAVLIB/LB:NUM24-LB:[32,1]SAVLIB/LB
SAV25:  .FCTR   SAVLIB/LB:NUM25-LB:[32,1]SAVLIB/LB
SAV26:  .FCTR   SAVLIB/LB:NUM26-LB:[32,1]SAVLIB/LB
SAV27:  .FCTR   SAVLIB/LB:NUM27-LB:[32,1]SAVLIB/LB
SAV29:  .FCTR   SAVLIB/LB:NUM29-LB:[32,1]SAVLIB/LB
SAV30:  .FCTR   SAVLIB/LB:NUM30-LB:[32,1]SAVLIB/LB
SAV31:  .FCTR   SAVLIB/LB:NUM31-LB:[32,1]SAVLIB/LB
SAV32:  .FCTR   SAVLIB/LB:NUM32-LB:[32,1]SAVLIB/LB
SAV33:  .FCTR   SAVLIB/LB:NUM33-LB:[32,1]SAVLIB/LB
SAV34:  .FCTR   SAVLIB/LB:NUM34-LB:[32,1]SAVLIB/LB
SAV35:  .FCTR   SAVLIB/LB:NUM35-LB:[32,1]SAVLIB/LB
SAV36:  .FCTR   SAVLIB/LB:NUM36-LB:[32,1]SAVLIB/LB
SAV37:  .FCTR   SAVLIB/LB:NUM37-LB:[32,1]SAVLIB/LB
SAV38:  .FCTR   SAVLIB/LB:NUM38-LB:[32,1]SAVLIB/LB
SAV39:  .FCTR   SAVLIB/LB:NUM39-LB:[32,1]SAVLIB/LB
SAV40:  .FCTR   SAVLIB/LB:NUM40-LB:[32,1]SAVLIB/LB
SAV41:  .FCTR   SAVLIB/LB:NUM41-LB:[32,1]SAVLIB/LB
SAV42:  .FCTR   SAVLIB/LB:NUM42-LB:[32,1]SAVLIB/LB
SAV43:  .FCTR   SAVLIB/LB:NUM43-LB:[32,1]SAVLIB/LB
SAV44:  .FCTR   SAVLIB/LB:NUM44-LB:[32,1]SAVLIB/LB
SAV45:  .FCTR   SAVLIB/LB:NUM45-LB:[32,1]SAVLIB/LB
SAV46:  .FCTR   SAVLIB/LB:NUM46-LB:[32,1]SAVLIB/LB
```

```
SAV47:   .FCTR    SAVLIB/LB:NUM47-LB:[32,1]SAVLIB/LB
SAV48:   .FCTR    SAVLIB/LB:NUM48-LB:[32,1]SAVLIB/LB
SAV49:   .FCTR    SAVLIB/LB:NUM49-LB:[32,1]SAVLIB/LB
SAV50:   .FCTR    SAVLIB/LB:NUM50-LB:[32,1]SAVLIB/LB
SAV51:   .FCTR    SAVLIB/LB:NUM51-LB:[32,1]SAVLIB/LB
LIB:     .FCTR    LB:[1,1]COROTS/LB
LIB2:    .FCTR    LB:[32,1]SAVLIB/LB:MOD49:MOD12:MOD13
         .END
```

APPENDIX F2

LIBCMP.CMD

```
COR LIB00,LIB00/-SP=LIB00
COR LIB01,LIB01/-SP=LIB01
COR LIB02,LIB02/-SP=LIB02
COR LIB03,LIB03/-SP=LIB03
COR LIB04,LIB04/-SP=LIB04
COR LIB05,LIB05/-SP=LIB05
COR LIB06,LIB06/-SP=LIB06
COR LIB07,LIB07/-SP=LIB07
COR LIB08,LIB08/-SP=LIB08
COR LIB09,LIB09/-SP=LIB09
COR LIB10,LIB10/-SP=LIB10
COR LIB11,LIB11/-SP=LIB11
COR LIB12,LIB12/-SP=LIB12
COR LIB13,LIB13/-SP=LIB13
COR LIB14,LIB14/-SP=LIB14
COR LIB15,LIB15/-SP=LIB15
COR LIB16,LIB16/-SP=LIB16
COR LIB17,LIB17/-SP=LIB17
COR LIB18,LIB18/-SP=LIB18
COR LIB19,LIB19/-SP=LIB19
COR LIB20,LIB20/-SP=LIB20
COR LIB21,LIB21/-SP=LIB21
COR LIB22,LIB22/-SP=LIB22
COR LIB23,LIB23/-SP=LIB23
COR LIB24,LIB24/-SP=LIB24
COR LIB25,LIB25/-SP=LIB25
COR LIB26,LIB26/-SP=LIB26
COR LIB27,LIB27/-SP=LIB27
COR LIB28,LIB28/-SP=LIB28
COR LIB29,LIB29/-SP=LIB29
COR LIB30,LIB30/-SP=LIB30
COR LIB31,LIB31/-SP=LIB31
COR LIB32,LIB32/-SP=LIB32
COR LIB33,LIB33/-SP=LIB33
COR LIB34,LIB34/-SP=LIB34
COR LIB35,LIB35/-SP=LIB35
COR LIB36,LIB36/-SP=LIB36
COR LIB37,LIB37/-SP=LIB37
COR LIB38,LIB38/-SP=LIB38
COR LIB39,LIB39/-SP=LIB39
COR LIB40,LIB40/-SP=LIB40
COR LIB41,LIB41/-SP=LIB41
COR LIB42,LIB42/-SP=LIB42
COR LIB43,LIB43/-SP=LIB43
COR LIB44,LIB44/-SP=LIB44
COR LIB45,LIB45/-SP=LIB45
COR LIB46,LIB46/-SP=LIB46
COR LIB47,LIB47/-SP=LIB47
COR LIB48,LIB48/-SP=LIB48
COR LIB49,LIB49/-SP=LIB49
COR LIB50,LIB50/-SP=LIB50
COR LIB51,LIB51/-SP=LIB51
COR LIB52,LIB52/-SP=LIB52
COR LIB53,LIB53/-SP=LIB53
COR LIB54,LIB54/-SP=LIB54
```

```
COR LIB55,LIB55/-SP=LIB55
COR LIB56,LIB56/-SP=LIB56
COR LIB57,LIB57/-SP=LIB57
COR LIB58,LIB58/-SP=LIB58
COR LIB59,LIB59/-SP=LIB59
COR LIB60,LIB60/-SP=LIB60
COR LIB61,LIB61/-SP=LIB61
COR LIB62,LIB62/-SP=LIB62
COR LIB63,LIB63/-SP=LIB63
COR LIB64,LIB64/-SP=LIB64
COR LIB65,LIB65/-SP=LIB65
COR LIB66,LIB66/-SP=LIB66
COR LIB67,LIB67/-SP=LIB67
```

APPENDIX.F3

OVLCMP.CMD

```
COR OVL00,OVL00/-SP=OVL00
COR OVL01,OVL01/-SP=OVL01
COR OVL02,OVL02/-SP=OVL02
COR OVL03,OVL03/-SP=OVL03
COR OVL04,OVL04/-SP=OVL04
COR OVL05,OVL05/-SP=OVL05
COR OVL06,OVL06/-SP=OVL06
COR OVL07,OVL07/-SP=OVL07
COR OVL08,OVL08/-SP=OVL08
COR OVL09,OVL09/-SP=OVL09
COR OVL10,OVL10/-SP=OVL10
COR OVL11,OVL11/-SP=OVL11
COR OVL12,OVL12/-SP=OVL12
COR OVL13,OVL13/-SP=OVL13
COR OVL14,OVL14/-SP=OVL14
COR OVL15,OVL15/-SP=OVL15
COR OVL16,OVL16/-SP=OVL16
COR OVL17,OVL17/-SP=OVL17
COR OVL18,OVL18/-SP=OVL18
COR OVL19,OVL19/-SP=OVL19
COR OVL20,OVL20/-SP=OVL20
COR OVL21,OVL21/-SP=OVL21
COR OVL22,OVL22/-SP=OVL22
COR OVL23,OVL23/-SP=OVL23
COR OVL24,OVL24/-SP=OVL24
COR OVL25,OVL25/-SP=OVL25
COR OVL26,OVL26/-SP=OVL26
COR OVL27,OVL27/-SP=OVL27
COR OVL28,OVL28/-SP=OVL28
COR OVL29,OVL29/-SP=OVL29
COR OVL30,OVL30/-SP=OVL30
COR OVL31,OVL31/-SP=OVL31
COR OVL32,OVL32/-SP=OVL32
COR OVL33,OVL33/-SP=OVL33
COR OVL34,OVL34/-SP=OVL34
COR OVL35,OVL35/-SP=OVL35
COR OVL36,OVL36/-SP=OVL36
COR OVL37,OVL37/-SP=OVL37
COR OVL38,OVL38/-SP=OVL38
COR OVL39,OVL39/-SP=OVL39
COR OVL40,OVL40/-SP=OVL40
COR OVL41,OVL41/-SP=OVL41
COR OVL42,OVL42/-SP=OVL42
COR OVL43,OVL43/-SP=OVL43
COR OVL44,OVL44/-SP=OVL44
COR OVL45,OVL45/-SP=OVL45
COR OVL46,OVL46/-SP=OVL46
COR OVL47,OVL47/-SP=OVL47
COR OVL48,OVL48/-SP=OVL48
COR OVL49,OVL49/-SP=OVL49
COR OVL50,OVL50/-SP=OVL50
COR OVL51,OVL51/-SP=OVL51
```

APPENDIX F4

LIBGEN.CMD

```
;THIS BUILDS THE LIBRARY FOR SAVANT
PIP SAVLIB.OLB;*/DE
LBR SAVLIB/CR
LBR SAVLIB/IN=LIB00
LBR SAVLIB/IN=LIB01
LBR SAVLIB/IN=LIB02
LBR SAVLIB/IN=LIB03
LBR SAVLIB/IN=LIB04
LBR SAVLIB/IN=LIB05
LBR SAVLIB/IN=LIB06
LBR SAVLIB/IN=LIB07
LBR SAVLIB/IN=LIB08
LBR SAVLIB/IN=LIB09
LBR SAVLIB/IN=LIB10
LBR SAVLIB/IN=LIB11
LBR SAVLIB/IN=LIB12
LBR SAVLIB/IN=LIB13
LBR SAVLIB/IN=LIB14
LBR SAVLIB/IN=LIB15
LBR SAVLIB/IN=LIB16
LBR SAVLIB/IN=LIB17
LBR SAVLIB/IN=LIB18
LBR SAVLIB/IN=LIB19
LBR SAVLIB/IN=LIB20
LBR SAVLIB/IN=LIB21
LBR SAVLIB/IN=LIB22
LBR SAVLIB/IN=LIB23
LBR SAVLIB/IN=LIB24
LBR SAVLIB/IN=LIB25
LBR SAVLIB/IN=LIB26
LBR SAVLIB/IN=LIB27
LBR SAVLIB/IN=LIB28
LBR SAVLIB/IN=LIB29
LBR SAVLIB/IN=LIB30
LBR SAVLIB/IN=LIB31
LBR SAVLIB/IN=LIB32
LBR SAVLIB/IN=LIB33
LBR SAVLIB/IN=LIB34
LBR SAVLIB/IN=LIB35
LBR SAVLIB/IN=LIB36
LBR SAVLIB/IN=LIB37
LBR SAVLIB/IN=LIB38
LBR SAVLIB/IN=LIB39
LBR SAVLIB/IN=LIB40
LBR SAVLIB/IN=LIB41
LBR SAVLIB/IN=LIB42
LBR SAVLIB/IN=LIB43
LBR SAVLIB/IN=LIB44
LBR SAVLIB/IN=LIB45
LBR SAVLIB/IN=LIB46
LBR SAVLIB/IN=LIB47
LBR SAVLIB/IN=LIB48
LBR SAVLIB/IN=LIB49
LBR SAVLIB/IN=LIB50
LBR SAVLIB/IN=LIB51
```

```
LBR SAVLIB/IN=LIB52
LBR SAVLIB/IN=LIB53
LBR SAVLIB/IN=LIB54
LBR SAVLIB/IN=LIB55
LBR SAVLIB/IN=LIB56
LBR SAVLIB/IN=LIB57
LBR SAVLIB/IN=LIB58
LBR SAVLIB/IN=LIB59
LBR SAVLIB/IN=LIB60
LBR SAVLIB/IN=LIB61
LBR SAVLIB/IN=LIB62
LBR SAVLIB/IN=LIB63
LBR SAVLIB/IN=LIB64
LBR SAVLIB/IN=LIB65
LBR SAVLIB/IN=LIB66
LBR SAVLIB/IN=LIB67
LBR SAVLIB/IN=OVL00
LBR SAVLIB/IN=OVL01
LBR SAVLIB/IN=OVL02
LBR SAVLIB/IN=OVL03
LBR SAVLIB/IN=OVL04
LBR SAVLIB/IN=OVL05
LBR SAVLIB/IN=OVL06
LBR SAVLIB/IN=OVL07
LBR SAVLIB/IN=OVL08
LBR SAVLIB/IN=OVL09
LBR SAVLIB/IN=OVL10
LBR SAVLIB/IN=OVL11
LBR SAVLIB/IN=OVL12
LBR SAVLIB/IN=OVL13
LBR SAVLIB/IN=OVL14
LBR SAVLIB/IN=OVL15
LBR SAVLIB/IN=OVL16
LBR SAVLIB/IN=OVL17
LBR SAVLIB/IN=OVL18
LBR SAVLIB/IN=OVL19
LBR SAVLIB/IN=OVL20
LBR SAVLIB/IN=OVL21
LBR SAVLIB/IN=OVL22
LBR SAVLIB/IN=OVL23
LBR SAVLIB/IN=OVL24
LBR SAVLIB/IN=OVL25
LBR SAVLIB/IN=OVL26
LBR SAVLIB/IN=OVL27
LBR SAVLIB/IN=OVL28
LBR SAVLIB/IN=OVL29
LBR SAVLIB/IN=OVL30
LBR SAVLIB/IN=OVL31
LBR SAVLIB/IN=OVL32
LBR SAVLIB/IN=OVL33
LBR SAVLIB/IN=OVL34
LBR SAVLIB/IN=OVL35
LBR SAVLIB/IN=OVL36
LBR SAVLIB/IN=OVL37
LBR SAVLIB/IN=OVL38
LBR SAVLIB/IN=OVL39
LBR SAVLIB/IN=OVL40
LBR SAVLIB/IN=OVL41
LBR SAVLIB/IN=OVL42
LBR SAVLIB/IN=OVL43
LBR SAVLIB/IN=OVL44
LBR SAVLIB/IN=OVL45
```

```
LBR SAVLIB/IN=OVL46
LBR SAVLIB/IN=OVL47
LBR SAVLIB/IN=OVL48
LBR SAVLIB/IN=OVL49
LBR SAVLIB/IN=OVL50
LBR SAVLIB/IN=OVL51
PIP /NV/CO=SAVLIB.OLB
PIP SAVLIB.OLB/PU
PIP *.OBJ;*/DE
COR SAVOVL,SAVOVL/-SP=SAVOVL
```

UNCLASSIFIED

APPENDIX F5

SAVBLD.CMD

```
SAVANT,SAVANT/CR/-SP=SAVANT/MP
MAXBUF=560
//
```


UNCLASSIFIED

UNCLASSIFIED

APPENDIX F6

SAVMN.CMD

```
;STARTING COMPILATION OF OVL??.COR FILES
@OVLCMP
;STARTING COMPILATION OF LIB??.COR FILES
@LIBCMP
;DELETE LISTING FILES
PIP *.LST;*/DE
;STARTING TO BUILD SAVANT LIBRARY
@LIBGEN
;STARTING TO TASK BUILD SAVANT
TKB @SAVBLD
;COMPLETION OF SAVANT BUILD
```

REPORT DOCUMENTATION PAGE

Overall security classification of this page

UNCLASSIFIED UNLIMITED

As far as possible this page should contain only unclassified information. If it is necessary to enter classified information, the box above must be marked to indicate the classification, e.g. Restricted, Confidential or Secret.

1. DRIC Reference (to be added by DRIC)	2. Originator's Reference	3. Agency Reference	4. Report Security Classification/Marking
	RAE TM FS(F) 452	N/A	UNCLASSIFIED UNLIMITED

5. DRIC Code for Originator	6. Originator (Corporate Author) Name and Location
7673000W	Royal Aircraft Establishment, Farnborough, Hants, UK
5a. Sponsoring Agency's Code	6a. Sponsoring Agency (Contract Authority) Name and Location
N/A	N/A

7. Title Modification of SAVANT to run on the PDP RSX-11M operating system.

7a. (For Translations) Title in Foreign Language

7b. (For Conference Papers) Title, Place and Date of Conference

8. Author 1, Surname, Initials	9a. Author 2	9b. Authors 3, 4 ...	10. Date	Pages	Refs.
Bluff, R.J.	Hay, P.T.		January 1982	110	4

11. Contract Number	12. Period	13. Project	14. Other Reference Nos.
N/A	N/A		

15. Distribution statement

(a) Controlled by - Head Flight Systems Department, RAE

(b) Special limitations (if any) -

16. Descriptors (Keywords) (Descriptors marked * are selected from TEST)

Avionics. Computer. System. Data bus.

17. Abstract

SAVANT (System Architecture Verification and Analysis Technique) is a computer program developed specifically to investigate the feasibility of an avionic system design, in terms of problems such as completeness and consistency.

The program was originally developed on a Prime 300 computer system. This memorandum describes alterations and modifications of SAVANT, to enable it to be used on a PDP 11 RSX-11M operating system.